
CCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCCCCCCCCCC OOOOOOOOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNNNNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNN NNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNNNNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCC OOO OOO NNN NNN VVV VVV
CCCCCCCCCCC OOOOOOOOO NNN NNN VVV
CCCCCCCCCCC OOOOOOOOO NNN NNN VVV
CCCCCCCCCCC OOOOOOOOO NNN NNN VVV

FILEID**CONVFSTLD

```
;    1          0001  0 %TITLE  'VAX-11 CONVERT'
;    2          0002  0 MODULE  CONV$FSTLD      ( IDENT='V04-000',
;    3          0003  0                         OPTLEVEL=3
;    4          0004  0                         ) =
;    5          0005  0
;    6          0006  1 BEGIN
;    7          0007  1
;    8          0008  1 !*****************************************************************************
;    9          0009  1 !*                                                                           *
;   10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;   11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;   12          0012  1 !*  ALL RIGHTS RESERVED.                                                     *
;   13          0013  1 !*                                                                           *
;   14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;   15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;   16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;   17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;   18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;   19          0019  1 !*  TRANSFERRED.                                                             *
;   20          0020  1 !*                                                                           *
;   21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;   22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;   23          0023  1 !*  CORPORATION.                                                             *
;   24          0024  1 !*                                                                           *
;   25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;   26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;   27          0027  1 !*                                                                           *
;   28          0028  1 !*                                                                           *
;   29          0029  1 !*****************************************************************************
```

```
;    31          0030  1 !++
;    32          0031  1 !
;    33          0032  1 ! Facility:     VAX-11 CONVERT
;    34          0033  1 !
;    35          0034  1 ! Abstract:     This module contains the high level calls for the fast load
;    36          0035  1 !               process along with the declaratons for the data specifically
;    37          0036  1 !               used by fast load
;    38          0037  1 !
;    39          0038  1 ! Contents:
;    40          0039  1 !               FAST_LOAD
;    41          0040  1 !               INIT_FAST_LOAD
;    42          0041  1 !               LOAD_PRIMARY
;    43          0042  1 !               LOAD_SECONDARY
;    44          0043  1 !               LOAD_DATA_BUCKET
;    45          0044  1 !               LOAD_INDEX_BUCKET
;    46          0045  1 !               FINISH_INDEX
;    47          0046  1 !               BACKUP_INDEX
;    48          0047  1 !
;    49          0048  1 ! Environment:
;    50          0049  1 !
;    51          0050  1 !               VAX/VMS Operating System
;    52          0051  1 !
;    53          0052  1 !--
;    54          0053  1
;    55          0054  1 !
;    56          0055  1 ! Author:       Keith B Thompson        Creation date:  August-1980
;    57          0056  1 !
;    58          0057  1 !
;    59          0058  1 ! Modified by:
;    60          0059  1 !
;    61          0060  1 !       V03-013 RAS0305         Ron Schaefer            7-May-1984
;    62          0061  1 !               Fix check for maximum index level so that we report
;    63          0062  1 !               an error rather than get an access violation if the
;    64          0063  1 !               index level exceeds 31.
;    65          0064  1 !
;    66          0065  1 !       V03-012 JWT0177         Jim Teague              17-Apr-1984
;    67          0066  1 !               CONVERT always tried to load a sidr bucket, even if
;    68          0067  1 !               all records in the file had null keys for the
;    69          0068  1 !               index, thereby corrupting the file.  Correct this
;    70          0069  1 !               error by making sure at least one non-null key is
;    71          0070  1 !               encountered for an index before allocating and
;    72          0071  1 !               loading a SIDR bucket.
;    73          0072  1 !
;    74          0073  1 !       V03-011 JWT0143         Jim Teague              25-Nov-1983
;    75          0074  1 !               CONVERT used to blindly add records until the fill
;    76          0075  1 !               factor was exceeded.  Now, check to see if adding
;    77          0076  1 !               a record will bring us closer to the fill factor.
;    78          0077  1 !               If we're closer before the addition (even though
;    79          0078  1 !               we may be short of the fill factor), then don't
;    80          0079  1 !               add the record.
;    81          0080  1 !
;    82          0081  1 !       V03-010 KBT0476         Keith B. Thompson       29-Jan-1983
;    83          0082  1 !               Add support for the ADD_KEY function
;    84          0083  1 !
;    85          0084  1 !       V03-009 KBT0459         Keith B. Thompson       10-Jan-1983
;    86          0085  1 !               Fix a bug when loading p3 sidrs with no dups
;    87          0086  1 !
```

```
;    88          0087  1 !        V03-008 KBT0404         Keith B. Thompson       19-Nov-1982
;    89          0088  1 !                Fix some of the sidr code
;    90          0089  1 !
;    91          0090  1 !        V03-007 KBT0382         Keith B. Thompson       26-Oct-1982
;    92          0091  1 !                Add prologue 3 sidr support
;    93          0092  1 !
;    94          0093  1 !        V03-006 KBT0375         Keith B. Thompson       20-Oct-1982
;    95          0094  1 !                Check for keys out of order from split_data
;    96          0095  1 !
;    97          0096  1 !        V03-005 KBT0349         Keith B. Thompson       4-Oct-1982
;    98          0097  1 !                Use new linkage definitions
;    99          0098  1 !
;   100          0099  1 !        V03-004 KBT0050         Keith Thompson          10-May-1982
;   101          0100  1 !                Check for empty file before calling finish index
;   102          0101  1 !
;   103          0102  1 !        V03-003 KBT0047         Keith Thompson          14-Apr-1982
;   104          0103  1 !                Fix end condition problem with the index buckets
;   105          0104  1 !
;   106          0105  1 !        V03-002 KBT0022         Keith Thompson          24-Mar-1982
;   107          0106  1 !                Fix problem with last data bucket being continuation bucket
;   108          0107  1 !                and more duplicate problems.  Change some linkages.
;   109          0108  1 !
;   110          0109  1 !        V03-001 KBT0012         Keith Thompson          16-Mar-1982
;   111          0110  1 !                Fix some prologue 3 duplicate bugs in load_data_bucket
;   112          0111  1 !                and remove prologue 3 secondary key code
;   113          0112  1 !****
```

```
;  115          0113  1
;  116          0114  1 PSECT
;  117          0115  1         OWN     = _CONV$FAST_D  (PIC),
;  118          0116  1         GLOBAL  = _CONV$FAST_D  (PIC),
;  119          0117  1         PLIT    = _CONV$PLIT    (SHARE,PIC),
;  120          0118  1         CODE    = _CONV$FAST_S  (SHARE,PIC);
;  121          0119  1
;  122          0120  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;  123          0121  1 LIBRARY 'SRC$:CONVERT';
;  124          0122  1
;  125          0123  1 DEFINE_ERROR_CODES;
;  126          0124  1
;  127          0125  1 EXTERNAL ROUTINE
;  128          0126  1         CONV$$GET_VM            : CL$GET_VM,
;  129          0127  1         CONV$$GET_TEMP_VM       : CL$GET_TEMP_VM,
;  130          0128  1         CONV$$FREE_TEMP_VM      : CL$FREE_TEMP_VM       NOVALUE,
;  131          0129  1         CONV$$EXCEPTION,
;  132          0130  1         CONV$$END_OF_FILE       : NOVALUE,
;  133          0131  1         CONV$$SORT_SECONDARY    : CL$SORT_SECONDARY,
;  134          0132  1         CONV$$GET_RECORD        : CL$GET_RECORD,
;  135          0133  1         CONV$$CHECK_S_DUP       : CL$JSB_REG_9,
;  136          0134  1         CONV$$CHECK_NULL        : CL$JSB_REG_9,
;  137          0135  1         CONV$$SPLIT_DATA        : CL$JSB_REG_9,
;  138          0136  1         CONV$$COMPRESS_KEY      : CL$JSB_REG_9  NOVALUE,
;  139          0137  1         CONV$$COMPRESS_INDEX    : CL$JSB_REG_9  NOVALUE,
;  140          0138  1         CONV$$MAKE_INDEX        : CL$JSB_REG_9  NOVALUE,
;  141          0139  1         CONV$$WRITE_VBN         : CL$JSB_REG_9  NOVALUE,
;  142          0140  1         CONV$$COPY_REY          : CL$COPY_KEY   NOVALUE,
;  143          0141  1         CONV$$WRITE_BUCKET      : CL$JSB_REG_9  NOVALUE,
;  144          0142  1         CONV$$GET_BUCKET        : CL$JSB_REG_9  NOVALUE,
;  145          0143  1         CONV$$INIT_BUCKET       : CL$JSB_REG_9  NOVALUE,
;  146          0144  1         CONV$$CREATE_HIGH_KEY   : CL$JSB_REG_9  NOVALUE,
;  147          0145  1         CONV$$WRITE_PROLOGUE    : NOVALUE,
;  148          0146  1         CONV$$CONVERT_VBN_ID    : CL$CONVERT_VBN_ID     NOVALUE,
;  149          0147  1         CONV$$SET_KEY_DESC      : CL$SET_KEY_DESC,
;  150          0148  1         CONV$$GET_NEXT_KEY      : CL$GET_NEXT_KEY,
;  151          0149  1         CONV$$WRITE_KEY_DESC    : CL$WRITE_KEY_DESC     NOVALUE;
;  152          0150  1
;  153          0151  1 FORWARD ROUTINE
;  154          0152  1         CONV$$INIT_FAST_LOAD    : CL$INIT_FAST_LOAD     NOVALUE,
;  155          0153  1         LOAD_PRIMARY            : CL$JSB_REG_9,
;  156          0154  1         CONV$$LOAD_SECONDARY    : CL$LOAD_SECONDARY     NOVALUE,
;  157          0155  1         LOAD_DATA_BUCKET        : CL$JSB_REG_8  NOVALUE,
;  158          0156  1         LOAD_INDEX_BUCKET       : CL$JSB_REG_9  NOVALUE,
;  159          0157  1         FINISH_INDEX            : CL$JSB_REG_9  NOVALUE,
;  160          0158  1         BACKUP_INDEX            : CL$JSB_REG_9  NOVALUE;
;  161          0159  1
;  162          0160  1 EXTERNAL
;  163          0161  1         CONV$GL_FILL            : LONG,
;  164          0162  1
;  165          0163  1         CONV$GW_OUT_REC_SIZ     : SIGNED WORD,          ! Output Rec. Size
;  166          0164  1
;  167          0165  1         CONV$GL_RECORD_COUNT,
;  168          0166  1         CONV$GL_EXCEPT_COUNT,
;  169          0167  1         CONV$GL_VALID_COUNT,
;  170          0168  1
;  171          0169  1         CONV$GW_MAX_REC_SIZ     : WORD,                 ! Aprox. size of record buffer
```

```
;  172          0170  1          CONV$GL_RFA_BUFFER,
;  173          0171  1
;  174          0172  1          CONV$AB_IN_RAB          : $RAB_DECL,
;  175          0173  1          CONV$AB_OUT_XABSUM      : $XABSUM_DECL,
;  176          0174  1          CONV$AB_OUT_FAB         : $FAB_DECL,
;  177          0175  1          CONV$AB_OUT_RAB         : $RAB_DECL,
;  178          0176  1          CONV$AB_RFA_RAB         : $RAB_DECL,
;  179          0177  1
;  180          0178  1          CONV$GL_EOF_VBN         : LONG,
;  181          0179  1          CONV$GB_PROL_V1         : BYTE,
;  182          0180  1          CONV$GB_PROL_V2         : BYTE,
;  183          0181  1          CONV$GB_PROL_V3         : BYTE,
;  184          0182  1          CONV$AR_PROLOGUE        : REF BLOCK [ ,BYTE ],
;  185          0183  1          CONV$AR_AREA_BLOCK      : REF BLOCKVECTOR [ ,AREA$C_BLN,BYTE ];
;  186          0184  1
;  187          0185  1 LITERAL
;  188          0186  1          FALSE = 0,
;  189          0187  1          TRUE  = 1;
;  190          0188  1
;  191          0189  1 MACRO
;  192          0190  1 !        Some needed macros to define the data record for a bucket
;  193          0191  1 !
;  194          0192  1          IRC$L_RRV_VBN   = 3,0,32,0%,    ! RRV VBN Pointer
;  195          0193  1          IRC$L_RRV_VBN3  = 5,0,32,0%,    ! RRV VBN Pointer (Prologue 3)
;  196          0194  1          IRC$W_VAR_SIZ   = 7,0,16,0%,    ! Var. Rec. Format Size field
;  197          0195  1          IRC$L_DUPCOUNT  = 2,0,32,0%,    ! Duplicate count field
;  198          0196  1          IRC$W_DUPSZ     = 6,0,16,0%,    ! Size field when dup. are allowed
;  199          0197  1          IRC$W_NODUPSZ   = 2,0,16,0%,    ! Size field when dup. are not allowed
;  200          0198  1          IRC$W_P3SZ      = 0,0,16,0%,    ! Size field for prologue 3 files
;  201          0199  1
;  202          0200  1 !        These macros make the code look a little better
;  203          0201  1 !
;  204          0202  1          BKT$W_VBNFS     = .CONV$GW_VBN_FS_PTR,0,16,0%,  ! VBN Freespace Pointer in index level
;  205          0203  1          BKT$W_VBNFS0    = .CONV$GW_VBN_FS_PTR0,0,16,0%, ! VBN Freespace Pointer at the data level
;  206          0204  1          BKT$L_LCBPTR    = .CONV$GW_LCB_PTR,0,32,0%;     ! Last Contuation Bucket Pointer
;  207          0205  1
;  208          0206  1 !        Data Decl. for Fast Load routines
;  209          0207  1 !
;  210          0208  1 GLOBAL
;  211          0209  1          CONV$GL_RECORD_PTR      : LONG,         ! Pointer to record bffer
;  212          0210  1
;  213          0211  1          CONV$GW_VBN_FS_PTR      : WORD,
;  214          0212  1          CONV$GW_VBN_FS_PTR0     : WORD,
;  215          0213  1          CONV$GW_LCB_PTR         : WORD,
;  216          0214  1
;  217          0215  1          CONV$GL_CTX_BLOCK       : LONG,         ! Pointer to the contex block
;  218          0216  1          CONV$GL_DUP_BUF         : LONG;         ! Pointer to the Duplicate buffer
;  219          0217  1
;  220          0218  1 OWN
;  221          0219  1          CONTINUATION    : BYTE,         ! Continuation bucket
;  222          0220  1          DUPLICATE       : BYTE SIGNED,  ! Duplicate record
;  223          0221  1
;  224          0222  1          SAVE_FREESPACE  : WORD,         ! Save pointer for backing up index
;  225          0223  1          SAVE_KEYFRESPC  : WORD,         !   "
;  226          0224  1          SAVE_VBNFS      : WORD;         !   "
;  227          0225  1
```

```
;   229          0226  1 %SBTTL  'FAST_LOAD'
;   230          0227  1 GLOBAL ROUTINE  CONV$$FAST_LOAD : CL$JSB_REG_11 =
;   231          0228  1 !++
;   232          0229  1 !
;   233          0230  1 ! Functional Description:
;   234          0231  1 !
;   235          0232  1 !       FAST_LOAD is the driving routine for the fast loading process.  It
;   236          0233  1 !       will load the primary key then sort and load all secondary keys if
;   237          0234  1 !       any.
;   238          0235  1 !
;   239          0236  1 ! Calling Sequence:
;   240          0237  1 !
;   241          0238  1 !       CONV$$FAST_LOAD()
;   242          0239  1 !
;   243          0240  1 ! Input Parameters:
;   244          0241  1 !       none
;   245          0242  1 !
;   246          0243  1 ! Implicit Inputs:
;   247          0244  1 !       none
;   248          0245  1 !
;   249          0246  1 ! Output Parameters:
;   250          0247  1 !       none
;   251          0248  1 !
;   252          0249  1 ! Implicit Outputs:
;   253          0250  1 !       none
;   254          0251  1 !
;   255          0252  1 ! Routine Value:
;   256          0253  1 !
;   257          0254  1 !       RMS$_EOF or error code
;   258          0255  1 !
;   259          0256  1 ! Routines called:
;   260          0257  1 !
;   261          0258  1 !       CONV$$INIT_FAST_LOAD
;   262          0259  1 !       LOAD_PRIMARY
;   263          0260  1 !       CONV$$END_OF_FILE
;   264          0261  1 !       CONV$$WRITE_PROLOGUE
;   265          0262  1 !       CONV$$SET_KEY_DESC
;   266          0263  1 !       CONV$$SORT_SECONDARY
;   267          0264  1 !       CONV$$LOAD_SECONDARY
;   268          0265  1 !       CONV$$WRITE_KEY_DESC
;   269          0266  1 !
;   270          0267  1 ! Side Effects:
;   271          0268  1 !       none
;   272          0269  1 !
;   273          0270  1 !--
;   274          0271  1
;   275          0272  2     BEGIN
;   276          0273  2
;   277          0274  2     DEFINE_KEY_DESC;
;   278          0275  2     DEFINE_CTX_GLOBAL;
;   279          0276  2     DEFINE_BUCRET_GLOBAL;
;   280          0277  2
;   281          0278  2     ! Init the fast load process for all keys
;   282          0279  2     !
;   283          0280  2     CONV$$INIT_FAST_LOAD( 0 );
;   284          0281  2
;   285          0282  2     ! Load the primary data and index
```

```
;   286         0283  2     !
;   287         0284  2     RET_ON_ERROR( LOAD_PRIMARY() );
;   288         0285  2
;   289         0286  2     ! Write prologue
;   290         0287  2     !
;   291         0288  2     CONV$$WRITE_PROLOGUE();
;   292         0289  2
;   293         0290  2     ! Also write the key desc.
;   294         0291  2     !
;   295         0292  2     CONV$$WRITE_KEY_DESC();
;   296         0293  2
;   297         0294  2     ! Finish off the input file
;   298         0295  2     !
;   299         0296  2     CONV$$END_OF_FILE();
;   300         0297  2
;   301         0298  2     ! Free the space taken up by the loading
;   302         0299  2     !
;   303         0300  2     CONV$$FREE_TEMP_VM();
;   304         0301  2
;   305         0302  2     ! Process the secondary keys if there we records put into the
;   306         0303  2     ! output file.
;   307         0304  2     !
;   308         0305  2     ! NOTE: This could cause secondary key indexes to be uninitialized.
;   309         0306  2     ! At the moment RMS doesn't mind, if they ever do, something must be fixed.
;   310         0307  2     !
;   311         0308  2     IF .CONV$GL_VALID_COUNT GTRU 0
;   312         0309  2     THEN
;   313         0310  2
;   314         0311  2         ! Loop for each key
;   315         0312  2         !
;   316         0313  2         WHILE CONV$$GET_NEXT_KEY()
;   317         0314  2         DO
;   318         0315  3             BEGIN
;   319         0316  3
;   320         0317  3             ! Set up the sort for the secondary key.  The sort is a INDEX sort.
;   321         0318  3             ! This type of sort will produce a file of RFA's and keys of the
;   322         0319  3             ! primary data level we just made.
;   323         0320  3             !
;   324         0321  3             RET_ON_ERROR( CONV$$SORT_SECONDARY() );
;   325         0322  3
;   326         0323  3             ! Now that the file is sorted get the data and load it in.
;   327         0324  3             !
;   328         0325  3             CONV$$LOAD_SECONDARY();
;   329         0326  3
;   330         0327  3             ! Write the prologue
;   331         0328  3             !
;   332         0329  3             CONV$$WRITE_PROLOGUE();
;   333         0330  3
;   334         0331  3             ! And the key descriptor
;   335         0332  3             !
;   336         0333  3             CONV$$WRITE_KEY_DESC();
;   337         0334  3
;   338         0335  3             ! Free the space taken up by the last key load
;   339         0336  3             !
;   340         0337  3             CONV$$FREE_TEMP_VM()
;   341         0338  3
;   342         0339  2             END;
```

```
;  343          0340  2
;  344          0341  2      RETURN RMS$_EOF
;  345          0342  2
;  346          0343  1      END;


                                        .TITLE  CONV$FSTLD VAX-11 CONVERT
                                        .IDENT  \V04-000\

                                        .PSECT  _CONV$FAST_D,NOEXE,  PIC,2

00000 CONV$GL_RECORD_PTR::
                                        .BLKB   4
00004 CONV$GW_VBN_FS_PTR::
                                        .BLKB   2
00006 CONV$GW_VBN_FS_PTR0::
                                        .BLKB   2
00008 CONV$GW_LCB_PTR::
                                        .BLKB   2
0000A                                   .BLKB   2
0000C CONV$GL_CTX_BLOCK::
                                        .BLKB   4
00010 CONV$GL_DUP_BUF::
                                        .BLKB   4
00014 CONTINUATION:
                                        .BLKB   1
00015 DUPLICATE:
                                        .BLKB   1
00016 SAVE_FREESPACE:
                                        .BLKB   2
00018 SAVE_KEYFRESPC:
                                        .BLKB   2
0001A SAVE_VBNFS:
                                        .BLKB   2

                                        .EXTRN  CONVERT$_FACILITY
                                        .EXTRN  CONV$_FAO_MAX, CONV$_BADBLK
                                        .EXTRN  CONV$_BADLOGIC, CONV$_BADSORT
                                        .EXTRN  CONV$_CONFQUAL, CONV$_CREATEDSTM
                                        .EXTRN  CONV$_CREA_ERR, CONV$_DELPRI
                                        .EXTRN  CONV$_DUP, CONV$_EXTN_ERR
                                        .EXTRN  CONV$_FATALEXC, CONV$_FILLIM
                                        .EXTRN  CONV$_IDX_LIM, CONV$_ILL_KEY
                                        .EXTRN  CONV$_ILL_VALUE
                                        .EXTRN  CONV$_INP_FILES
                                        .EXTRN  CONV$_INSVIRMEM
                                        .EXTRN  CONV$_INVBKT, CONV$_KEY
                                        .EXTRN  CONV$_KEYREF, CONV$_LOADIDX
                                        .EXTRN  CONV$_NARG, CONV$_NI
                                        .EXTRN  CONV$_NOKEY, CONV$_NOTIDX
                                        .EXTRN  CONV$_NOTSEQ, CONV$_NOWILD
                                        .EXTRN  CONV$_ORDER, CONV$_OPENEXC
                                        .EXTRN  CONV$_OPENIN, CONV$_OPENOUT
                                        .EXTRN  CONV$_PAD, CONV$_PLV
                                        .EXTRN  CONV$_PROERR, CONV$_PROL_WRT
                                        .EXTRN  CONV$_READERR, CONV$_RSK
                                        .EXTRN  CONV$_RSZ, CONV$_RTL
```

```
                                        .EXTRN  CONV$_RTS, CONV$_SEQ
                                        .EXTRN  CONV$_UDF_BKS, CONV$_UDF_BLK
                                        .EXTRN  CONV$_VFC, CONV$_WRITEERR
                                        .EXTRN  CONV$$GET_VM, CONV$$GET_TEMP_VM
                                        .EXTRN  CONV$$FREE_TEMP_VM
                                        .EXTRN  CONV$$EXCEPTION
                                        .EXTRN  CONV$$END_OF_FILE
                                        .EXTRN  CONV$$SORT_SECONDARY
                                        .EXTRN  CONV$$GET_RECORD
                                        .EXTRN  CONV$$CHECK_S_DUP
                                        .EXTRN  CONV$$CHECK_NULL
                                        .EXTRN  CONV$$SPLIT_DATA
                                        .EXTRN  CONV$$COMPRESS_KEY
                                        .EXTRN  CONV$$COMPRESS_INDEX
                                        .EXTRN  CONV$$MAKE_INDEX
                                        .EXTRN  CONV$$WRITE_VBN
                                        .EXTRN  CONV$$COPY_REY, CONV$$WRITE_BUCKET
                                        .EXTRN  CONV$$GET_BUCKET
                                        .EXTRN  CONV$$INIT_BUCKET
                                        .EXTRN  CONV$$CREATE_HIGH_KEY
                                        .EXTRN  CONV$$WRITE_PROLOGUE
                                        .EXTRN  CONV$$CONVERT_VBN_ID
                                        .EXTRN  CONV$$SET_KEY_DESC
                                        .EXTRN  CONV$$GET_NEXT_KEY
                                        .EXTRN  CONV$$WRITE_KEY_DESC
                                        .EXTRN  CONV$GL_FILL, CONV$GW_OUT_REC_SIZ
                                        .EXTRN  CONV$GL_RECORD_COUNT
                                        .EXTRN  CONV$GL_EXCEPT_COUNT
                                        .EXTRN  CONV$GL_VALID_COUNT
                                        .EXTRN  CONV$GW_MAX_REC_SIZ
                                        .EXTRN  CONV$GL_RFA_BUFFER
                                        .EXTRN  CONV$AB_IN_RAB, CONV$AB_OUT_XABSUM
                                        .EXTRN  CONV$AB_OUT_FAB
                                        .EXTRN  CONV$AB_OUT_RAB
                                        .EXTRN  CONV$AB_RFA_RAB
                                        .EXTRN  CONV$GL_EOF_VBN
                                        .EXTRN  CONV$GB_PROC_V1
                                        .EXTRN  CONV$GB_PROL_V2
                                        .EXTRN  CONV$GB_PROL_V3
                                        .EXTRN  CONV$AR_PROLOGUE
                                        .EXTRN  CONV$AR_AREA_BLOCK

                                        .PSECT  _CONV$FAST_S,NOWRT,  SHR,  PIC,2

                    7E      59  7D 00000 CONV$$FAST_LOAD::
                                         MOVQ    R9, -(SP)                          ; 0227
                            7E  D4 00003         CLRL    -(SP)                      ; 0280
                        0000V   30 00005         BSBW    CONV$$INIT_FAST_LOAD       ;
                    5E      04  C0 00008         ADDL2   #4, SP                     ;
                        0000V   30 0000B         BSBW    LOAD_PRIMARY               ; 0284
                    39      50  E9 0000E         BLBC    STATUS, 3$                 ;
            0000G   CF      00  FB 00011         CALLS   #0, CONV$$WRITE_PROLOGUE   ; 0288
                        0000G   30 00016         BSBW    CONV$$WRITE_KEY_DESC       ; 0292
            0000G   CF      00  FB 00019         CALLS   #0, CONV$$END_OF_FILE      ; 0296
                        0000G   30 0001E         BSBW    CONV$$FREE_TEMP_VM         ; 0300
                    0000G   CF  D5 00021         TSTL    CONV$GL_VALID_COUNT        ; 0308
                            1C  13 00025         BEQL    2$                         ;
```

```
                                0000G 30 00027 1$:      BSBW    CONV$$GET_NEXT_KEY                      ; 0313
                          16          50 E9 0002A           BLBC    R0, 2$                                  ;
                                0000G 30 0002D           BSBW    CONV$$SORT_SECONDARY                    ; 0321
                          17          50 E9 00030           BLBC    STATUS, 3$                              ;
                                0000V 30 00033           BSBW    CONV$$LOAD_SECONDARY                    ; 0325
                    0000G CF          00 FB 00036           CALLS   #0, CONV$$WRITE_PROLOGUE                ; 0329
                                0000G 30 0003B           BSBW    CONV$$WRITE_KEY_DESC                    ; 0333
                                0000G 30 0003E           BSBW    CONV$$FREE_TEMP_VM                      ; 0337
                                   E4 11 00041           BRB     1$                                      ;
                          50 0001827A 8F DO 00043 2$:      MOVL    #98938, R0                              ; 0341
                          59          8E 7D 0004A 3$:      MOVQ    (SP)+, R9                               ; 0343
                                      05 0004D              RSB                                             ;

; Routine Size:  78 bytes,    Routine Base:  _CONV$FAST_S + 0000
```

```
;   348          0344  1 %SBTTL  'INIT_FAST_LOAD'
;   349          0345  1 GLOBAL ROUTINE  CONV$$INIT_FAST_LOAD ( MAX_KEY ) : CL$INIT_FAST_LOAD NOVALUE =
;   350          0346  1 !++
;   351          0347  1 !
;   352          0348  1 ! Functional Description:
;   353          0349  1 !
;   354          0350  1 !       Initialize the fast load process.  Get memory for buffers and set up
;   355          0351  1 !       pointers.  There are up to 3 pointers to record buffers at each level
;   356          0352  1 !       RCP, RDP and LKB for level 0 and prologue 3 files.  The RCP, LKP and
;   357          0353  1 !       RDP for all but level 0 pointers are set here.  The proper sizes are:
;   358          0354  1 !
;   359          0355  1 !                                       ----------------------------------------
;   360          0356  1 !       LEVEL 0         RCP --->!       max_key + 13                           !
;   361          0357  1 !                               ----------------------------------------
;   362          0358  1 !
;   363          0359  1 !                               ----------------------------------------
;   364          0360  1 !       LEVEL 1+        RCP --->!       5                                      !
;   365          0361  1 !                               ----------------------------------------
;   366          0362  1 !
;   367          0363  1 !                               ----------------------------------------
;   368          0364  1 !                       RDP --->!       max_key + 2                            !
;   369          0365  1 !                               ----------------------------------------
;   370          0366  1 !
;   371          0367  1 !                               ----------------------------------------
;   372          0368  1 !       ALL LEVELS      LKP --->!       max_key                                !
;   373          0369  1 !                               ----------------------------------------
;   374          0370  1 !
;   375          0371  1 !       The RDP for level 0 is set in load_primary and load_secondary.
;   376          0372  1 !
;   377          0373  1 ! Calling Sequence:
;   378          0374  1 !
;   379          0375  1 !       INIT_FAST_LOAD();
;   380          0376  1 !
;   381          0377  1 ! Input Parameters:
;   382          0378  1 !       none
;   383          0379  1 !
;   384          0380  1 ! Implicit Inputs:
;   385          0381  1 !       none
;   386          0382  1 !
;   387          0383  1 ! Output Parameters:
;   388          0384  1 !       none
;   389          0385  1 !
;   390          0386  1 ! Implicit Outputs:
;   391          0387  1 !       none
;   392          0388  1 !
;   393          0389  1 ! Routine Value:
;   394          0390  1 !       none
;   395          0391  1 !
;   396          0392  1 ! Routines Called:
;   397          0393  1 !
;   398          0394  1 !       CONV$$SET_KEY_DESC
;   399          0395  1 !       CONV$$GET_NEXT_KEY
;   400          0396  1 !       CONV$GET_VM
;   401          0397  1 !
;   402          0398  1 ! Side Effects:
;   403          0399  1 !
;   404          0400  1 !       Sets the end of file VBN pointer.  Allocates memory for record buffers.
```

```
;  405          0401  1 !       Sets up the record data pointers, record control pointers and last key
;  406          0402  1 !       pointers.
;  407          0403  1 !
;  408          0404  1 !--
;  409          0405  1
;  410          0406  2     BEGIN
;  411          0407  2
;  412          0408  2     DEFINE_CTX;
;  413          0409  2     DEFINE_BUCKET;
;  414          0410  2     DEFINE_KEY_DESC;
;  415          0411  2
;  416          0412  2     ! Since we are doing block IO we dont need the XABs anymore
;  417          0413  2     ! If they are keep around area xabs (if any) will override the fab during
;  418          0414  2     ! an extend a screw everything up.
;  419          0415  2     !
;  420          0416  2     CONV$AB_OUT_FAB [ FAB$L_XAB ] = 0;
;  421          0417  2
;  422          0418  2     ! Find the end of file VBN. In a new file it the one block past the last
;  423          0419  2     ! allocated area (the last area may not be allocated therefore look at
;  424          0420  2     ! one)
;  425          0421  2     !
;  426          0422  2     BEGIN                   ! HIGH_VBN local
;  427          0423  2
;  428          0424  2     LOCAL       HIGH_VBN;
;  429          0425  2
;  430          0426  2     HIGH_VBN = 0;
;  431          0427  2
;  432          0428  2     INCR AREA FROM 0 TO ( .CONV$AB_OUT_XABSUM [ XAB$B_NOA ] - 1 ) BY 1
;  433          0429  2     DO
;  434          0430  2
;  435          0431  3         ! If the current extent starts at a higher VBN then the last one
;  436          0432  3         ! us this one to find the end of file
;  437          0433  3         !
;  438          0434  3         IF .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CVBN ] GTR .HIGH_VBN
;  439          0435  3         THEN
;  440          0436  3
;  441          0437  4             ! The end of file is this the start of this extent plus the number
;  442          0438  4             ! of blocks in the extent
;  443          0439  4             !
;  444          0440  4             CONV$GL_EOF_VBN = .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CVBN ] +
;  445          0441  4                               .CONV$AR_AREA_BLOCK [ .AREA,AREA$L_CNBLK ];
;  446          0442  4
;  447          0443  2     END;                    ! HIGH_VBN local
;  448          0444  2
;  449          0445  2     ! Calculate the max space needed for index key buffers and init. the contex
;  450          0446  2     ! block.  If it was not given.
;  451          0447  2     !
;  452          0448  2     IF .MAX_KEY EQLU 0
;  453          0449  2     THEN
;  454          0450  3         BEGIN
;  455          0451  3
;  456          0452  3         ! The max. size is the size of the logest key.
;  457          0453  3         ! So we check each key desc.
;  458          0454  3         !
;  459          0455  3         CONV$$SET_KEY_DESC( 0 );
;  460          0456  3
;  461          0457  3         DO
```

```
;  462          0458  1          }
;  463          0459  1                  IF .KEY_DESC [ KEY$B_KEYSZ ] GTR .MAX_KEY
;  464          0460  1                  THEN
;  465          0461  1                      MAX_KEY = .KEY_DESC [ KEY$B_KEYSZ ]
;  466          0462  1
;  467          0463  1              UNTIL NOT CONV$$GET_NEXT_KEY()
;  468          0464  1              END;
;  469          0465  1
;  470          0466  1          BEGIN
;  471          0467  1
;  472          0468  1          LOCAL       BYTES;
;  473          0469  1
;  474          0470  1          ! Figure the total number of bytes. (SEE ABOVE)
;  475          0471  1          !
;  476          0472  1          BYTES = ( .MAX_KEY * ( MAX_IDX_LVL + 1 ) ) + ( MAX_IDX_LVL * 7 ) + 13;
;  477          0473  1
;  478          0474  1          ! For Prologue 3 files we may need the last key buffers
;  479          0475  1          !
;  480          0476  1          IF .CONV$GB_PROL_V3
;  481          0477  1          THEN
;  482          0478  1              BYTES = .BYTES + ( .MAX_KEY * ( MAX_IDX_LVL - 1 ) );
;  483          0479  1
;  484          0480  1          ! Add the space for the contex block
;  485          0481  1          !
;  486          0482  1          BYTES = .BYTES + ( MAX_IDX_LVL * CTX$K_BLN );
;  487          0483  1
;  488          0484  1          ! Get the zero filled space
;  489          0485  1          !
;  490          0486  1          CONV$GL_CTX_BLOCK = CONV$$GET_VM ( .BYTES )
;  491          0487  1
;  492          0488  1          END;
;  493          0489  1
;  494          0490  1          ! Set all of the record control pointers and record data pointers for
;  495          0491  1          ! level one (1) and above.
;  496          0492  1          !
;  497          0493  1          CTX = .CONV$GL_CTX_BLOCK;
;  498          0494  1
;  499          0495  1          CTX [ CTX$L_RCP ] = .CTX + ( MAX_IDX_LVL * CTX$K_BLN );
;  500          0496  1
;  501          0497  2          BEGIN       ! BUFFER_OFFSET local
;  502          0498  2
;  503          0499  2          LOCAL       BUFFER_OFFSET;
;  504          0500  2
;  505          0501  2          BUFFER_OFFSET = .CTX [ CTX$L_RCP ] + .MAX_KEY + 13;
;  506          0502  2
;  507          0503  2          INCR I FROM 1 TO ( MAX_IDX_LVL - 1 ) BY 1
;  508          0504  2          DO
;  509          0505  3              BEGIN
;  510          0506  3              CTX = .CTX + CTX$K_BLN;
;  511          0507  3              CTX [ CTX$B_LEVEL ] = .I;
;  512          0508  3              CTX [ CTX$L_RCP ] = .BUFFER_OFFSET;
;  513          0509  3              CTX [ CTX$L_RDP ] = .BUFFER_OFFSET + 5;
;  514          0510  3              BUFFER_OFFSET = .BUFFER_OFFSET + .MAX_KEY + 7
;  515          0511  2              END;
;  516          0512  2
;  517          0513  2          ! Set up the last key buffer for level 0
;  518          0514  2          !
```

```
;  519     0515  3     CTX = .CONV$GL_CTX_BLOCK;
;  520     0516  3     CTX [ CTX$L_LKP ] = .BUFFER_OFFSET;
;  521     0517  3
;  522     0518  3     ! Set up the last key buffer if necessary for levels 1 and above
;  523     0519  3     !
;  524     0520  3     IF .CONV$GB_PROL_V3
;  525     0521  3     THEN
;  526     0522  3         INCR I FROM 1 TO ( MAX_IDX_LVL - 1 ) BY 1
;  527     0523  3         DO
;  528     0524  4             BEGIN
;  529     0525  4             CTX = .CTX + CTX$K_BLN;
;  530     0526  4             BUFFER_OFFSET = .BUFFER_OFFSET + .MAX_KEY;
;  531     0527  4             CTX [ CTX$L_LKP ] = .BUFFER_OFFSET
;  532     0528  4             END
;  533     0529  4
;  534     0530  2     END;        ! BUFFER_OFFSET local
;  535     0531  2
;  536     0532  2     RETURN
;  537     0533  2
;  538     0534  1     END;


                                        1C  BB 00000 CONV$$INIT_FAST_LOAD::
                                                     PUSHR   #^M<R2,R3,R4>                          ; 0345
                                0000G   CF  D4 00002         CLRL    CONV$AB_OUT_FAB+36             ; 0416
                                        54  D4 00006         CLRL    HIGH_VBN                       ; 0426
                                53      0000G CF 9A 00008    MOVZBL  CONV$AB_OUT_XABSUM+8, R3        ; 0428
                                50          01  CE 0000D    MNEGL   #1, AREA                       ; 0434
                                            1D  11 00010     BRB     2$
                            52              50  06 78 00012 1$:     ASHL    #6, AREA, R2
                            51              52  0000G CF C1 00016    ADDL3   CONV$AR_AREA_BLOCK, R2, R1
                                            54  0C  A1 D1 0001C    CMPL    12(R1), HIGH_VBN
                                            0D  15 00020         BLEQ    2$
                                            52  0000G CF C0 00022    ADDL2   CONV$AR_AREA_BLOCK, R2  ; 0441
                    0000G   CF      0C      A1  10  A2 C1 00027    ADDL3   16(R2), 12(RT), CONV$GL_EOF_VBN
                            DF              50      53 F2 0002F 2$:  AOBLSS  R3, AREA, 1$           ; 0434
                                        10      AE  D5 00033     TSTL    MAX_KEY                    ; 0448
                                            1C  12 00036         BNEQ    5$
                                            7E  D4 00038         CLRL    -(SP)                      ; 0455
                                        0000G   30 0003A         BSBW    CONV$$SET_KEY_DESC
                                5E              04  C0 0003D    ADDL2   #4, SP
10  AE      14  AB                  08          00  ED 00040 3$:     CMPZV   #0, #8, 20(KEY_DESC), MAX_KEY  ; 0459
                                            05  15 00047         BLEQ    4$
                        10  AE      14  AB      9A 00049         MOVZBL  20(KEY_DESC), MAX_KEY      ; 0461
                                        0000G   30 0004E 4$:     BSBW    CONV$$GET_NEXT_KEY         ; 0463
                                EC          50  E8 00051         BLBS    R0, 3$
                                52      10  AE  D0 00054 5$:     MOVL    MAX_KEY, R2                ; 0472
                            51              52  21 C5 00058    MULL3   #33, R2, R1
                                50      00ED    C1 9E 0005C    MOVAB   237(R1), BYTES
                                07      0000G   CF E9 00061    BLBC    CONV$GB_PROL_V3, 6$        ; 0476
                            51              52  1F C5 00066    MULL3   #31, R2, R1                ; 0478
                                50          51  C0 0006A     ADDL2   R1, BYTES
                                50      0B80    C0 9E 0006D 6$:     MOVAB   2944(R0), BYTES        ; 0482
                                            50  DD 00072         PUSHL   BYTES                      ; 0486
```

```
                                 0000G 30 00074          BSBW    CONV$$GET_VM                                    ;
                             5E    04 C0 00077          ADDL2   #4, SP                                          ;
                      0000'  CF    50 D0 0007A          MOVL    R0, CONV$GL_CTX_BLOCK                           ;
                             5A 0000' CF D0 0007F          MOVL    CONV$GL_CTX_BLOCK, CTX                          ; 0493
                          30 AA 0B80  CA 9E 00084          MOVAB   2944(R10), 48(CTX)                              ; 0495
                    50    52   30  AA C1 0008A          ADDL3   48(CTX), R2, R0                                 ; 0501
                          50    0D  C0 0008F          ADDL2   #13, BUFFER_OFFSET                              ;
                          51    01  D0 00092          MOVL    #1, I                                           ; 0503
                          5A    5C  AA 9E 00095 7$:     MOVAB   92(R10), CTX                                    ; 0506
                       02 AA    51  90 00099          MOVB    I, 2(CTX)                                       ; 0507
                       30 AA    50  D0 0009D          MOVL    BUFFER_OFFSET, 48(CTX)                          ; 0508
                       34 AA 05  A0 9E 000A1          MOVAB   5(R0), 52(CTX)                                  ; 0509
                          50 07 A240 9E 000A6          MOVAB   7(R2)[BUFFER_OFFSET], BUFFER_OFFSET             ; 0510
                    E6    51    1F  F3 000AB          AOBLEQ  #31, I, 7$                                      ;
                          5A 0000' CF D0 000AF          MOVL    CONV$GL_CTX_BLOCK, CTX                          ; 0515
                       3C AA    50  D0 000B4          MOVL    BUFFER_OFFSET, 60(CTX)                          ; 0516
                          12 0000G CF E9 000B8          BLBC    CONV$GB_PROL_V3, 9$                             ; 0520
                          51    01  D0 000BD          MOVL    #1, I                                           ; 0522
                          5A    5C  AA 9E 000C0 8$:     MOVAB   92(R10), CTX                                    ; 0525
                          50    52  C0 000C4          ADDL2   R2, BUFFER_OFFSET                               ; 0526
                       3C AA    50  D0 000C7          MOVL    BUFFER_OFFSET, 60(CTX)                          ; 0527
                    F1    51    1F  F3 000CB          AOBLEQ  #31, I, 8$                                      ;
                                1C  BA 000CF 9$:      POPR    #^M<R2,R3,R4>                                   ; 0534
                                05 000D1              RSB                                                     ;

; Routine Size:  210 bytes,    Routine Base:  _CONV$FAST_S + 004E
```

```
;   540          0535  1 %SBTTL  'LOAD_PRIMARY'
;   541          0536  1 ROUTINE LOAD_PRIMARY : CL$JSB_REG_9 =
;   542          0537  1 !++
;   543          0538  1 !
;   544          0539  1 ! Functional Description:
;   545          0540  1 !
;   546          0541  1 !       Loads the primary key of a index sequential file.
;   547          0542  1 !
;   548          0543  1 ! Calling Sequence:
;   549          0544  1 !
;   550          0545  1 !       LOAD_PRIMARY()
;   551          0546  1 !
;   552          0547  1 ! Input Parameters:
;   553          0548  1 !       none
;   554          0549  1 !
;   555          0550  1 ! Implicit Inputs:
;   556          0551  1 !       none
;   557          0552  1 !
;   558          0553  1 ! Output Parameters:
;   559          0554  1 !       none
;   560          0555  1 !
;   561          0556  1 ! Implicit Outputs:
;   562          0557  1 !       none
;   563          0558  1 !
;   564          0559  1 ! Routine Value:
;   565          0560  1 !
;   566          0561  1 !       RMS$_EOF or error codes
;   567          0562  1 !
;   568          0563  1 ! Routine Called:
;   569          0564  1 !
;   570          0565  1 !       CONV$$SET_KEY_DESC
;   571          0566  1 !       CONV$$GET_TEMP_VM
;   572          0567  1 !       CONV$$GET_BUCKET
;   573          0568  1 !       CONV$$GET_RECORD
;   574          0569  1 !       CONV$$EXCEPTION
;   575          0570  1 !       CONV$$SPLIT_DATA
;   576          0571  1 !       LOAD_DATA_BUCKET
;   577          0572  1 !       FINISH_INDEX
;   578          0573  1 !
;   579          0574  1 ! Side Effects:
;   580          0575  1 !
;   581          0576  1 !       Loads primary key
;   582          0577  1 !
;   583          0578  1 !--
;   584          0579  1
;   585          0580  2     BEGIN
;   586          0581  2
;   587          0582  2     LABEL
;   588          0583  2         DUP_BLK;
;   589          0584  2
;   590          0585  2     DEFINE_CTX;
;   591          0586  2     DEFINE_BUCKET;
;   592          0587  2     DEFINE_KEY_DESC;
;   593          0588  2
;   594          0589  2     CTX = .CONV$GL_CTX_BLOCK;
;   595          0590  2
;   596          0591  2     ! Set key to the primary index
```

```
;  597          0592  2      !
;  598          0593  2      CONV$$SET_KEY_DESC( 0 );
;  599          0594  2
;  600          0595  2      ! Errors on the rab from now on are WRITEERRs
;  601          0596  2      !
;  602          0597  2      CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;
;  603          0598  2
;  604          0599  2      ! For prologue 3 files we need an extra buffer for the data record
;  605          0600  2      ! Else we let the REC_DATA_PTR point to the user buffer of the output rab
;  606          0601  2      !
;  607          0602  2      IF .CONV$GB_PROL_V3
;  608          0603  2      THEN
;  609          0604  3          BEGIN
;  610          0605  3
;  611          0606  3          LOCAL   BYTES;
;  612          0607  3
;  613          0608  3          ! The worst case is fully non compressed record with compression info
;  614          0609  3          !
;  615          0610  3          BYTES = .CONV$GW_MAX_REC_SIZ + 3;
;  616          0611  3
;  617          0612  3          ! Get the space for the data buffer
;  618          0613  3          !
;  619          0614  3          ! Record data pointer at level 0 will point to the new buffer
;  620          0615  3          !
;  621          0616  3          CTX [ CTX$L_RDP ] = CONV$$GET_TEMP_VM ( .BYTES )
;  622          0617  3
;  623          0618  3          END
;  624          0619  2      ELSE
;  625          0620  2
;  626          0621  2          ! Record data pointer at level 0 points to Record Ptr
;  627          0622  2          !
;  628          0623  2          CTX [ CTX$L_RDP ] = .CONV$GL_RECORD_PTR;
;  629          0624  2
;  630          0625  2      ! Get the Buckets for the data area and at least the first level of the index
;  631          0626  2      !
;  632          0627  2      ! Get the bucket for level 0
;  633          0628  2      !
;  634          0629  2      CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_DANUM ] );
;  635          0630  2
;  636          0631  2      KEY_DESC [ KEY$L_LDVBN ] = .CTX [ CTX$L_CURRENT_VBN ];
;  637          0632  2
;  638          0633  2      ! Get the bucket for level 1
;  639          0634  2      !
;  640          0635  2      CTX = .CTX + CTX$K_BLN;
;  641          0636  2      CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_LANUM ] );
;  642          0637  2
;  643          0638  2      CTX = .CONV$GL_CTX_BLOCK;
;  644          0639  2
;  645          0640  2      ! For the primary key the Data comes from GET_RECORD. NOTE: Don't use the
;  646          0641  2      ! UBF of the input RAB since some record conversion may be done. Also note
;  647          0642  2      ! the RBF pointer of the output RAB is destroyed after the first call to
;  648          0643  2      ! WRITE_BUCKET but it is ok to use it now.
;  649          0644  2      !
;  650          0645  3      BEGIN
;  651          0646  3
;  652          0647  3      DEFINE_RECORD_CTRL_GLOBAL;
;  653          0648  3
```

```
;  654          0649  3     LOCAL
;  655          0650  3         STATUS;
;  656          0651  3
;  657          0652  3     RECORD_CTRL = .CTX [ CTX$L_RCP ];
;  658          0653  3
;  659          0654  3     ! Main record processing loop.  The call to GET_RECORD does any record format
;  660          0655  3     ! processing and exception handling before it returns.  The size of the record
;  661          0656  3     ! is passed back by OUT_REC_SIZ.
;  662          0657  3     !
;  663          0658  4     WHILE ( STATUS = CONV$$GET_RECORD() )
;  664          0659  3     DO
;  665          0660  4         BEGIN           ! Main Loop
;  666          0661  4
;  667          0662  4 DUP_BLK:
;  668          0663  5         BEGIN           ! DUP_BLK Primary duplicate block
;  669          0664  5
;  670          0665  5         ! If the record is shorter the minium record length of the primary key
;  671          0666  5         ! cause an exception
;  672          0667  5         !
;  673          0668  5         IF .CONV$GW_OUT_REC_SIZ LSS .KEY_DESC [ KEY$W_MINRECSZ ]
;  674          0669  5         THEN
;  675          0670  6             BEGIN
;  676          0671  6
;  677          0672  6             LOCAL STATUS;
;  678          0673  6
;  679          0674  6             ! If it was not fatal continue else exit
;  680          0675  6             !
;  681          0676  6             IF STATUS = CONV$$EXCEPTION( CONV$_RSK )
;  682          0677  6             THEN
;  683          0678  6                 LEAVE DUP_BLK
;  684          0679  6             ELSE
;  685          0680  6                 RETURN .STATUS
;  686          0681  5             END;
;  687          0682  5
;  688          0683  5         ! Seperate the key from the data record if necessary and do
;  689          0684  5         ! data compression if necessary also check if this is a duplicate
;  690          0685  5         ! or the key is out of order
;  691          0686  5         !
;  692          0687  5         DUPLICATE = CONV$$SPLIT_DATA();
;  693          0688  5
;  694          0689  5         ! If out of order, i.e. duplicate = -1, signal exception and continue
;  695          0690  5         !
;  696          0691  5         IF .DUPLICATE LSS 0
;  697          0692  5         THEN
;  698          0693  6             BEGIN
;  699          0694  6
;  700          0695  6             LOCAL       STATUS;
;  701          0696  6
;  702          0697  6             ! If not fatal exception then continue else bomb
;  703          0698  6             !
;  704          0699  6             IF STATUS = CONV$$EXCEPTION ( CONV$_SEQ )
;  705          0700  6             THEN
;  706          0701  6                 LEAVE DUP_BLK
;  707          0702  6             ELSE
;  708          0703  6                 RETURN .STATUS
;  709          0704  6
;  710          0705  5             END;
```

```
;  711          0706  5
;  712          0707  5          ! If we got a dup and we dont allow dups then cause an exception
;  713          0708  5          !
;  714          0709  6          IF .DUPLICATE AND ( NOT .KEY_DESC [ KEY$V_DUPKEYS ] )
;  715          0710  5          THEN
;  716          0711  6              BEGIN
;  717          0712  6
;  718          0713  6              LOCAL       STATUS;
;  719          0714  6
;  720          0715  6              ! If not fatal exception then continue else bomb
;  721          0716  6              !
;  722          0717  6              IF STATUS = CONV$$EXCEPTION ( CONV$_DUP )
;  723          0718  6              THEN
;  724          0719  6                  LEAVE DUP_BLK
;  725          0720  6              ELSE
;  726          0721  6                  RETURN .STATUS
;  727          0722  6
;  728          0723  5              END;
;  729          0724  5
;  730          0725  5          ! Set up the control byte for the record
;  731          0726  5          !
;  732          0727  5          RECORD_CTRL [ IRC$B_CONTROL ] = 2;
;  733          0728  5
;  734          0729  5          ! Set the size field int the record
;  735          0730  5          !
;  736          0731  5          IF .CONV$GB_PROL_V3
;  737          0732  5          THEN
;  738          0733  6              BEGIN
;  739          0734  6
;  740          0735  6              ! A small non compressed fixed length record has no size field
;  741          0736  6              !
;  742          0737  6              IF .KEY_DESC [ KEY$V_REC_COMPR ] OR
;  743          0738  6                  .KEY_DESC [ KEY$V_KEY_COMPR ] OR
;  744          0739  7                  ( .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQLU FAB$C_VAR )
;  745          0740  6              THEN
;  746          0741  6                  RECORD_CTRL [ 9,0,16,0 ] = .CTX [ CTX$W_RCS ] +
;  747          0742  6                                             .CTX [ CTX$W_RDS ] - 11
;  748          0743  6              END
;  749          0744  5          ELSE
;  750          0745  5
;  751          0746  5              ! Set up the record size for var. length records
;  752          0747  5              ! for prologue 1 and 2 files
;  753          0748  5              !
;  754          0749  5              IF .CONV$AB_OUT_FAB [ FAB$B_RFM ] EQLU FAB$C_VAR
;  755          0750  5              THEN
;  756          0751  5                  RECORD_CTRL [ IRC$W_VAR_SIZ ] = .CONV$GW_OUT_REC_SIZ;
;  757          0752  5
;  758          0753  5          ! If we are in a continuation bucket and the current record is NOT a
;  759          0754  5          ! duplicate then write the current bucket out and start a new one
;  760          0755  5          !
;  761          0756  5          ! For optimumization do the continuation check first
;  762          0757  5          !
;  763          0758  5          IF .CONTINUATION THEN IF NOT .DUPLICATE
;  764          0759  5          THEN
;  765          0760  6              BEGIN
;  766          0761  6
;  767          0762  6              CONV$$WRITE_BUCKET();
```

```
;  768          0763  6
;  769          0764  6              CONV$$INIT_BUCKET();
;  770          0765  6
;  771          0766  6              ! Continuation no longer need be set.  (the next record will always
;  772          0767  6              ! fix into the new bucket
;  773          0768  6              !
;  774          0769  6              CONTINUATION = _CLEAR
;  775          0770  6
;  776          0771  6              END;
;  777          0772  5
;  778          0773  5          ! Load the record
;  779          0774  5          !
;  780          0775  5          LOAD_DATA_BUCKET()
;  781          0776  5
;  782          0777  5          END                 ! DUP_BLK Primary duplicate block
;  783          0778  5          END;                ! Main loop
;  784          0779  4
;  785          0780  4      ! If we exited because of end of file and there are records in the file
;  786          0781  4      ! then finish off the index
;  787          0782  4      !
;  788          0783  4      IF ( .STATUS EQLU RMS$_EOF ) AND
;  789          0784  4                           ( .CONV$GL_RECORD_COUNT NEQU .CONV$GL_EXCEPT_COUNT )
;  790          0785  4      THEN
;  791          0786  4          FINISH_INDEX();
;  792          0787  4
;  793          0788  4      RETURN CONV$_SUCCESS
;  794          0789  4
;  795          0790  4      END
;  796          0791  1      END;


                               0104   8F   BB 00000 LOAD_PRIMARY:
                                                          PUSHR   #^M<R2,R8>                              ; 0536
                           5A      0000'  CF   D0 00004          MOVL    CONV$GL_CTX_BLOCK, CTX          ; 0589
                                          7E   D4 00009          CLRL    -(SP)                           ; 0593
                                    0000G 30   0000B             BSBW    CONV$$SET_KEY_DESC              ;
                           5E             04   C0 0000E          ADDL2   #4, SP                          ;
                     0000G CF  00000000G  8F   D0 00011          MOVL    #CONV$_WRITEERR, CONV$AB_OUT_RAB+24 ; 0597
                           16      0000G  CF   E9 0001A          BLBC    CONV$GB_PROL_V3, 1$             ; 0602
                           50      0000G  CF   3C 0001F          MOVZWL  CONV$GW_MAX_REC_SIZ, BYTES      ; 0610
                           50             03   C0 00024          ADDL2   #3, BYTES                       ;
                                          50   DD 00027          PUSHL   BYTES                           ; 0616
                                    0000G 30   00029             BSBW    CONV$$GET_TEMP_VM               ;
                           5E             04   C0 0002C          ADDL2   #4, SP                          ;
                        34 AA             50   D0 0002F          MOVL    R0, 52(CTX)                     ;
                                          06   11 00033          BRB     2$                              ;
                        34 AA      0000'  CF   D0 00035 1$:       MOVL    CONV$GL_RECORD_PTR, 52(CTX)     ; 0623
                           7E        08   AB   9A 0003B 2$:       MOVZBL  8(KEY_DESC), -(SP)              ; 0629
                                    0000G 30   0003F             BSBW    CONV$$GET_BUCKET                ;
                        54 AB        08   AA   D0 00042          MOVL    8(CTX), @4(KEY_DESC)            ; 0631
                           5A        5C   AA   9E 00047          MOVAB   92(R10), CTX                    ; 0635
                           6E        07   AB   9A 0004B          MOVZBL  7(KEY_DESC), (SP)               ; 0636
                                    0000G 30   0004F             BSBW    CONV$$GET_BUCKET                ;
                           5E             04   C0 00052          ADDL2   #4, SP                          ;
```

```
                                 5A      0000'  CF   D0 00055          MOVL    CONV$GL_CTX_BLOCK, CTX                                  ; 0638
                                 58        30   AA   D0 0005A          MOVL    48(CTX), RECORD_CTRL                                    ; 0652
                                          0000G 30   0005E 3$:        BSBW    CONV$$GET_RECORD                                        ; 0658
                                 52             50   D0 00061          MOVL    R0, STATUS                                              ;
                                 03             52   E8 00064          BLBS    STATUS, 4$                                              ;
                                              0091 31   00067          BRW     14$                                                     ;
                                 50        16   AB   3C 0006A 4$:      MOVZWL  22(KEY_DESC), R0                                        ; 0668
        50      0000G  CF        10             00   EC 0006E          CMPV    #0, #16, CONV$GW_OUT_REC_SIZ, R0                        ;
                                                08   18 00075          BGEQ    5$                                                      ;
                                    00000000G   8F   DD 00077          PUSHL   #CONV$_RSK                                              ; 0676
                                                21   11 0007D          BRB     7$                                                      ;
                                              0000G 30   0007F 5$:      BSBW    CONV$$SPLIT_DATA                                        ; 0687
                           0000'  CF                50   90 00082          MOVB    R0, DUPLICATE                                           ;
                                                08   18 00087          BGEQ    6$                                                      ; 0691
                                    00000000G   8F   DD 00089          PUSHL   #CONV$_SEQ                                              ; 0699
                                                0F   11 0008F          BRB     7$                                                      ;
                                 14      0000'  CF   E9 00091 6$:      BLBC    DUPLICATE, 9$                                           ; 0709
                                 10        10   AB   E8 00096          BLBS    16(KEY_DESC), 9$                                        ;
                                    00000000G   8F   DD 0009A          PUSHL   #CONV$_DUP                                              ; 0717
                           0000G  CF                01   FB 000A0 7$:      CALLS   #1, CONV$$EXCEPTION                                     ;
                                 6B             50   E9 000A5          BLBC    STATUS, 16$                                             ;
                                                B4   11 000A8 8$:      BRB     3$                                                      ; 0719
                                 68             02   90 000AA 9$:      MOVB    #2, (RECORD_CTRL)                                       ; 0727
                                 23      0000G  CF   E9 000AD          BLBC    CONV$GB_PROC_V3, 11$                                    ; 0731
                                   10           AB   95 000B2          TSTB    16(KEY_DESC)                                            ; 0737
                                                0C   19 000B5          BLSS    10$                                                     ;
                        07         10     AB        06   E0 000B7          BBS     #6, 16(KEY_DESC), 10$                                   ; 0738
                                 02      0000G  CF   91 000BC          CMPB    CONV$AB_OUT_FAB+31, #2                                  ; 0739
                                                1F   12 000C1          BNEQ    12$                                                     ;
                                 50        38   AA   3C 000C3 10$:      MOVZWL  56(CTX), R0                                             ; 0742
                                 51        3A   AA   3C 000C7          MOVZWL  58(CTX), R1                                             ;
                                 50             51   C0 000CB          ADDL2   R1, R0                                                  ;
                     09    A8                  50   A3 000CE          SUBW3   #11, R0, 9(RECORD_CTRL)                                 ;
                                                0D   11 000D3          BRB     12$                                                     ; 0733
                                 02      0000G  CF   91 000D5 11$:      CMPB    CONV$AB_OUT_FAB+31, #2                                  ; 0749
                                                06   12 000DA          BNEQ    12$                                                     ;
                              07    A8        0000G  CF   B0 000DC          MOVW    CONV$GW_OUT_REC_SIZ, 7(RECORD_CTRL)                     ; 0751
                                 0F      0000'  CF   E9 000E2 12$:      BLBC    CONTINUATION, 13$                                       ; 0758
                                 0A      0000'  CF   E8 000E7          BLBS    DUPLICATE, 13$                                          ;
                                              0000G 30   000EC          BSBW    CONV$$WRITE_BUCKET                                      ; 0762
                                              0000G 30   000EF          BSBW    CONV$$INIT_BUCKET                                       ; 0764
                                       0000'  CF   94 000F2          CLRB    CONTINUATION                                            ; 0769
                                              0000V 30   000F6 13$:      BSBW    LOAD_DATA_BUCKET                                        ; 0775
                                                AD   11 000F9          BRB     8$                                                      ; 0660
                           0001827A   8F             52   D1 000FB 14$:      CMPL    STATUS, #98938                                          ; 0783
                                                0C   12 00102          BNEQ    15$                                                     ;
                              0000G  CF        0000G  CF   D1 00104          CMPL    CONV$GL_RECORD_COUNT, CONV$GL_EXCEPT_COUNT              ; 0784
                                                03   13 0010B          BEQL    15$                                                     ;
                                              0000V 30   0010D          BSBW    FINISH_INDEX                                            ; 0786
                                 50             01   D0 00110 15$:      MOVL    #1, R0                                                  ; 0788
                                       0104     8F   BA 00113 16$:      POPR    #^M<R2,R8>                                              ; 0791
                                                05   00117          RSB                                                             ;

; Routine Size:  280 bytes,    Routine Base:  _CONV$FAST_S + 0120

;  797          0792  1
```

```
;  799          0793  1 %SBTTL  'LOAD_SECONDARY'
;  800          0794  1 GLOBAL ROUTINE  CONV$$LOAD_SECONDARY : CL$LOAD_SECONDARY NOVALUE =
;  801          0795  1 !++
;  802          0796  1 !
;  803          0797  1 ! Functional Description:
;  804          0798  1 !
;  805          0799  1 !       Loads a secondary key of an index sequential file.  Which secondary
;  806          0800  1 !       index depends on KEY_REF.  The secondary
;  807          0801  1 !       data records are read from the RFA file created and opened by
;  808          0802  1 !       SORT_SECONDARY.  NOTE:  The overall operation of LOAD_SECONDARY is
;  809          0803  1 !       fundementally different then LOAD_PRIMARY.
;  810          0804  1 !
;  811          0805  1 ! Calling Sequence:
;  812          0806  1 !
;  813          0807  1 !       CONV$$LOAD_SECONDARY();
;  814          0808  1 !
;  815          0809  1 ! Input Parameters:
;  816          0810  1 !       none
;  817          0811  1 !
;  818          0812  1 ! Implicit Inputs:
;  819          0813  1 !       none
;  820          0814  1 !
;  821          0815  1 ! Output Parameters:
;  822          0816  1 !       none
;  823          0817  1 !
;  824          0818  1 ! Implicit Outputs:
;  825          0819  1 !       none
;  826          0820  1 !
;  827          0821  1 ! Routine Value:
;  828          0822  1 !
;  829          0823  1 !       RMS$_EOF or error codes
;  830          0824  1 !
;  831          0825  1 ! Routines Called:
;  832          0826  1 !
;  833          0827  1 !       CONV$$GET_BUCKET
;  834          0828  1 !       CONV$$GET_TEMP_VM
;  835          0829  1 !       $GET
;  836          0830  1 !       CONV$$CHECK_NULL
;  837          0831  1 !       CONV$$CHECK_S_DUP
;  838          0832  1 !       LOAD_DATA_BUCKET
;  839          0833  1 !       CONV$$COPY_KEY
;  840          0834  1 !       CONV$$WRITE_BUCKET
;  841          0835  1 !       CONV$$INIT_BUCKET
;  842          0836  1 !       CONV$$CONVERT_VBN_ID
;  843          0837  1 !       FINISH_INDEX
;  844          0838  1 !
;  845          0839  1 ! Side Effects:
;  846          0840  1 !
;  847          0841  1 !       Loads secondary index defined by KEY_REF
;  848          0842  1 !
;  849          0843  1 !--
;  850          0844  1
;  851          0845  2     BEGIN
;  852          0846  2
;  853          0847  2     DEFINE_CTX;
;  854          0848  2     DEFINE_BUCKET;
;  855          0849  2     DEFINE_KEY_DESC;
```

```
;   856          0850  2
;   857          0851  2      LABEL
;   858          0852  2          NULL_BLK;
;   859          0853  2
;   860          0854  2      LOCAL
;   861          0855  2          DUP_COUNT,
;   862          0856  2          MAX_NUM_DUP;
;   863          0857  2
;   864          0858  2      ! Init some values
;   865          0859  2      !
;   866          0860  2      CONTINUATION = _CLEAR;
;   867          0861  2      DUPLICATE = _CLEAR;
;   868          0862  2
;   869          0863  2      ! Errors on the rab from now on are WRITEERRs
;   870          0864  2      !
;   871          0865  2      CONV$AB_OUT_RAB [ RAB$L_CTX ] = CONV$_WRITEERR;
;   872          0866  2
;   873          0867  2      ! Point to the first block
;   874          0868  2      !
;   875          0869  2      CTX = .CONV$GL_CTX_BLOCK;
;   876          0870  2
;   877          0871  2      ! Get the Buckets for the secondary data area and at least the
;   878          0872  2      ! first level of the index
;   879          0873  2      !
;   880          0874  2      ! Get the bucket for level 0
;   881          0875  2      !
;   882          0876  2      CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_DANUM ] );
;   883          0877  2
;   884          0878  2      KEY_DESC [ KEY$L_LDVBN ] = .CTX [ CTX$L_CURRENT_VBN ];
;   885          0879  2
;   886          0880  2      ! Get the bucket for level 1
;   887          0881  2      !
;   888          0882  2      CTX = .CTX + CTX$K_BLN;
;   889          0883  2      CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_LANUM ] );
;   890          0884  2
;   891          0885  2      CTX = .CONV$GL_CTX_BLOCK;
;   892          0886  2
;   893          0887  2      ! Before we start we need to calcluate the size of the level 0 index record
;   894          0888  2      ! buffer.  This calculation is VERY important it must be very accurate!
;   895          0889  2      !
;   896          0890  2      ! If we allow dup. keys the it becomes complicated
;   897          0891  2      !
;   898          0892  2      ! Find out the max. number of duplicates that can fit in this bucket
;   899          0893  2      !
;   900          0894  2      IF .KEY_DESC [ KEY$V_DUPKEYS ]
;   901          0895  2      THEN
;   902          0896  2
;   903          0897  2          ! Sizes are different for prologue 3
;   904          0898  2          !
;   905          0899  2          IF .CONV$GB_PROL_V3
;   906          0900  2          THEN
;   907          0901  2
;   908          0902  2              ! For compression it is also different
;   909          0903  2              !
;   910          0904  2              IF .KEY_DESC [ KEY$V_IDX_COMPR ]
;   911          0905  2              THEN
;   912          0906  2
```

```
;  913          0907  2                    ! The space in the bucket minus the key size and the record
;  914          0908  2                    !       overhead (2+2) divided by the size of the SIDR record
;  915          0909  2                    !       pointer (7)
;  916          0910  2                    !
;  917          0911  2                    MAX_NUM_DUP = ( .CTX [ CTX$W_SPC ] -
;  918          0912  2                                  ( .KEY_DESC [ KEY$B_KEYSZ ] + 4 ) ) / 7
;  919          0913  2                ELSE
;  920          0914  2
;  921          0915  2                    ! The space in the bucket minus the key size and the record
;  922          0916  2                    !       overhead (2) divided by the size of the SIDR record
;  923          0917  2                    !       pointer (7)
;  924          0918  2                    !
;  925          0919  2                    MAX_NUM_DUP = ( .CTX [ CTX$W_SPC ] -
;  926          0920  2                                  ( .KEY_DESC [ KEY$B_KEYSZ ] + 2 ) ) / 7
;  927          0921  2
;  928          0922  2            ELSE
;  929          0923  2
;  930          0924  2                ! The space in the bucket minus the key size and the record
;  931          0925  2                !       overhead (8) divided by the size of the SIDR record
;  932          0926  2                !       pointer (6)
;  933          0927  2                !
;  934          0928  2                MAX_NUM_DUP = ( .CTX [ CTX$W_SPC ] -
;  935          0929  2                              ( .KEY_DESC [ KEY$B_KEYSZ ] + 8 ) ) / 6
;  936          0930  2        ELSE
;  937          0931  2            MAX_NUM_DUP = 1;
;  938          0932  2
;  939          0933  2        BEGIN
;  940          0934  2
;  941          0935  2        LOCAL       BYTES;
;  942          0936  2
;  943          0937  2        ! The size of the level 0 buffer consist of:
;  944          0938  2        !
;  945          0939  2        !   Space for all RRVs (one for each dup) : Largest rrv - prologue 3, 7 bytes
;  946          0940  2        !   Overhead : Maximun overhead - prologue 1, 8 bytes
;  947          0941  2        !
;  948          0942  2        ! We also need a duplicate buffer for things which is the size of the key
;  949          0943  2        !
;  950          0944  2        BYTES = ( .MAX_NUM_DUP * 7 ) + 8 + .KEY_DESC [ KEY$B_KEYSZ ];
;  951          0945  2
;  952          0946  2        ! Allocate the memory for the buffer
;  953          0947  2        ! The level 0 data record pointers points to this buffer
;  954          0948  2        !
;  955          0949  2        CTX [ CTX$L_RDP ] = CONV$$GET_TEMP_VM ( .BYTES );
;  956          0950  2
;  957          0951  2        ! The duplicate buffer is just past that
;  958          0952  2        !
;  959          0953  2        CONV$GL_DUP_BUF = .CTX [ CTX$L_RDP ] + ( .MAX_NUM_DUP * 7 ) + 8
;  960          0954  2
;  961          0955  2        END;
;  962          0956  2
;  963          0957  2        ! For the secondary key the Data comes from $GET on the RFA RAB
;  964          0958  2        !
;  965          0959  2        BEGIN                   ! RECORD_CTRL local
;  966          0960  2
;  967          0961  2        DEFINE_RECORD_CTRL_GLOBAL;
;  968          0962  2
;  969          0963  3        LOCAL
```

```
;  970     0964  3          ALL_NULL,
;  971     0965  3          SKIP,
;  972     0966  3          STATUS;
;  973     0967  3
;  974     0968  3      SKIP = _CLEAR;
;  975     0969  3
;  976     0970  3      RECORD_CTRL = .CTX [ CTX$L_RCP ];
;  977     0971  3
;  978     0972  3      ALL_NULL = _SET;     ! Could be nothing but null keys, you know...
;  979     0973  3
;  980     0974  3      ! Main record processing loop. The size of the record is returned in
;  981     0975  3      ! RFA_RAB [ RAB$W_RSZ ]
;  982     0976  3      !
;  983     0977  4      WHILE ( STATUS = $GET( RAB=CONV$AB_RFA_RAB ) )
;  984     0978  3      DO
;  985     0979  4          BEGIN            ! Main Loop
;  986     0980  4
;  987     0981  4  NULL_BLK:
;  988     0982  5          BEGIN            ! NULL_BLK null key value block
;  989     0983  5
;  990     0984  5          LOCAL   DUP;
;  991     0985  5
;  992     0986  5          ! If the record is too short (does not contain the complete key)
;  993     0987  5          ! then treat it as a null key
;  994     0988  5          !
;  995     0989  5          IF ( .CONV$AB_RFA_RAB [ RAB$W_RSZ ] - 6 ) LSSU .KEY_DESC [ KEY$B_KEYSZ ]
;  996     0990  5          THEN
;  997     0991  5              LEAVE NULL_BLK;
;  998     0992  5
;  999     0993  5          ! If the file allows null keys check to see if this is one
; 1000     0994  5          !
; 1001     0995  5          IF .KEY_DESC [ KEY$V_NULKEYS ]
; 1002     0996  5          THEN
; 1003     0997  5
; 1004     0998  5              ! If this is a null key then just ignore this record
; 1005     0999  5              !
; 1006     1000  5              IF CONV$$CHECK_NULL()
; 1007     1001  5              THEN
; 1008     1002  5                  LEAVE NULL_BLK;
; 1009     1003  5
; 1010     1004  5          !
; 1011     1005  5          ! If we got a non-null key, then all_null can no longer be true
; 1012     1006  5          !
; 1013     1007  5          IF .ALL_NULL THEN ALL_NULL = _CLEAR;
; 1014     1008  5
; 1015     1009  5          ! Check to see if this is a duplicate.
; 1016     1010  5          !
; 1017     1011  5          DUP = CONV$$CHECK_S_DUP();
; 1018     1012  5
; 1019     1013  5          ! Process the key
; 1020     1014  5          !
; 1021     1015  5          IF .KEY_DESC [ KEY$V_DUPKEYS ]
; 1022     1016  5          THEN
; 1023     1017  6              BEGIN
; 1024     1018  6
; 1025     1019  6              ! If this was a dup
; 1026     1020  6              !
```

```
;  1027          1021  6              IF .DUP
;  1028          1022  6              THEN
;  1029          1023  7                  BEGIN
;  1030          1024  7
;  1031          1025  7                  DUP_COUNT = .DUP_COUNT + 1;
;  1032          1026  7
;  1033          1027  7                  ! If we have exceeded the max number of dups per bucket then
;  1034          1028  7                  ! get rid of this bucket and start a new one
;  1035          1029  7                  !
;  1036          1030  7                  IF .DUP_COUNT GEQ .MAX_NUM_DUP
;  1037          1031  7                  THEN
;  1038          1032  8                      BEGIN
;  1039          1033  8
;  1040          1034  8                      LOAD_DATA_BUCKET();
;  1041          1035  8
;  1042          1036  8                      ! The record to go into the next bucket will be a duplicate
;  1043          1037  8                      !
;  1044          1038  8                      DUPLICATE = _SET;
;  1045          1039  8
;  1046          1040  8                      ! We are now in a continuation bucket
;  1047          1041  8                      !
;  1048          1042  8                      SKIP = _SET;
;  1049          1043  8
;  1050          1044  8                      ! Copy the key into the record (in a continuation bucket
;  1051          1045  8                      ! there is no dup count ie. the 4)
;  1052          1046  8                      !
;  1053          1047  8                      CONV$$COPY_KEY( 4 );
;  1054          1048  8
;  1055          1049  8                      ! Start counting dups again
;  1056          1050  8                      !
;  1057          1051  8                      DUP_COUNT = 0;
;  1058          1052  8
;  1059          1053  8                      ! Set the sidr array record size
;  1060          1054  8                      !
;  1061          1055  8                      CTX [ CTX$W_RDS ] = 0;
;  1062          1056  8
;  1063          1057  8                      ! Set some control fields. NOTE: COPY_KEY sets prologue 3
;  1064          1058  8                      ! record size field.
;  1065          1059  8                      !
;  1066          1060  8                      IF NOT .CONV$GB_PROL_V3
;  1067          1061  8                      THEN
;  1068          1062  9                          BEGIN
;  1069          1063  9
;  1070          1064  9                          ! A continuation record has no duplicate pointer
;  1071          1065  9                          !
;  1072          1066  9                          RECORD_CTRL [ IRC$B_CONTROL ] = IRC$M_NOPTRSZ;
;  1073          1067  9
;  1074          1068  9                          ! Prologue 1,2 size field includes a key
;  1075          1069  9                          !
;  1076          1070  9                          RECORD_CTRL [ IRC$W_NODUPSZ ] = .KEY_DESC [ KEY$B_KEYSZ ]
;  1077          1071  9
;  1078          1072  9                          END
;  1079          1073  8                      END
;  1080          1074  7                  END
;  1081          1075  6              ELSE
;  1082          1076  7                  BEGIN
;  1083          1077  7
```

```
;  1084     1078  7              ! If this is the first non-dup then don't load anything else
;  1085     1079  7              ! load the last record processed
;  1086     1080  7              !
;  1087     1081  7              IF NOT .CTX [ CTX$V_FST ]
;  1088     1082  7              THEN
;  1089     1083  8                  BEGIN
;  1090     1084  8
;  1091     1085  8                  LOAD_DATA_BUCKET();
;  1092     1086  8
;  1093     1087  8                  ! The next record will not be a duplicate record
;  1094     1088  8                  !
;  1095     1089  8                  DUPLICATE = _CLEAR;
;  1096     1090  8
;  1097     1091  8                  ! If we were in a continuatio bucket then dont make an index
;  1098     1092  8                  ! for it. Also write the bucket because we don't put anything
;  1099     1093  8                  ! in a bucket after a dup.
;  1100     1094  8                  !
;  1101     1095  8                  IF .SKIP
;  1102     1096  8                  THEN
;  1103     1097  9                      BEGIN
;  1104     1098  9                      SKIP = _CLEAR;
;  1105     1099  9
;  1106     1100  9                      CONV$$WRITE_BUCKET();
;  1107     1101  9                      CONV$$INIT_BUCKET();
;  1108     1102  9
;  1109     1103  9                      ! The next record will always fit into the new bucket
;  1110     1104  9                      ! so clearing the continuation flag is ok
;  1111     1105  9                      !
;  1112     1106  9                      CONTINUATION = _CLEAR
;  1113     1107  9
;  1114     1108  9                      END
;  1115     1109  7                  END;
;  1116     1110  7
;  1117     1111  7              ! Copy the key into the record past the dup count field (ie the 8)
;  1118     1112  7              !
;  1119     1113  7              CONV$$COPY_KEY( 8 );
;  1120     1114  7
;  1121     1115  7              ! Start counting the dups
;  1122     1116  7              !
;  1123     1117  7              DUP_COUNT = 0;
;  1124     1118  7
;  1125     1119  7              ! Set the sidr array record size
;  1126     1120  7              !
;  1127     1121  7              CTX [ CTX$W_RDS ] = 0;
;  1128     1122  7
;  1129     1123  7              ! Set some control fields. NOTE: COPY_KEY sets prologue 3
;  1130     1124  7              ! record size field.
;  1131     1125  7              !
;  1132     1126  7              IF NOT .CONV$GB_PROL_V3
;  1133     1127  7              THEN
;  1134     1128  8                  BEGIN
;  1135     1129  8
;  1136     1130  8                  ! The size of the dup pointer (1=4bytes)
;  1137     1131  8                  !
;  1138     1132  8                  RECORD_CTRL [ IRC$B_CONTROL ] = 1;
;  1139     1133  8
;  1140     1134  8                  ! Zero the field (not implemented)
```

```
;  1141     1135  8                    !
;  1142     1136  8                    RECORD_CTRL [ IRC$L_DUPCOUNT ] = 0;
;  1143     1137  8                    !
;  1144     1138  8                    ! Prologue 1,2 size field includes a key
;  1145     1139  8                    !
;  1146     1140  8                    RECORD_CTRL [ IRC$W_DUPSZ ] = .KEY_DESC [ KEY$B_KEYSZ ]
;  1147     1141  8                    !
;  1148     1142  8                    END
;  1149     1143  8
;  1150     1144  6                ;
;  1151     1145  6                !
;  1152     1146  6                ! Add to the size of the dup for this record.
;  1153     1147  6                !
;  1154     1148  7                IF .CONV$GB_PROL_V3
;  1155     1149  7                THEN
;  1156     1150  7                    !
;  1157     1151  7                    ! A prologue 3 RRV is 7 bytes (1 control,2 ID,4 VBN)
;  1158     1152  7                    !
;  1159     1153  7                    RECORD_CTRL [ IRC$W_P3SZ ] = .RECORD_CTRL [ IRC$W_P3SZ ] + 7
;  1160     1154  7                    !
;  1161     1155  7                ELSE
;  1162     1156  8                    BEGIN
;  1163     1157  8                    !
;  1164     1158  8                    ! A prologue 1,2 RRV is 6 bytes (1 control,1 ID,4 VBN )
;  1165     1159  8                    !
;  1166     1160  8                    IF .RECORD_CTRL [ IRC$V_NOPTRSZ ]
;  1167     1161  8                    THEN
;  1168     1162  8                        RECORD_CTRL [ IRC$W_NODUPSZ ] =
;  1169     1163  8                                            .RECORD_CTRL [ IRC$W_NODUPSZ ] + 6
;  1170     1164  8                    ELSE
;  1171     1165  8                        RECORD_CTRL [ IRC$W_DUPSZ ] =
;  1172     1166  8                                            .RECORD_CTRL [ IRC$W_DUPSZ ] + 6
;  1173     1167  7                    END )
;  1174     1168  7                    !
;  1175     1169  6                END
;  1176     1170  5            ELSE
;  1177     1171  6                BEGIN
;  1178     1172  6                !
;  1179     1173  6                ! If the keys are duplicate and we are not allowing dup. then error
;  1180     1174  6                !
;  1181     1175  6                IF .DUP
;  1182     1176  6                THEN
;  1183     1177  6                    SIGNAL_STOP( CONV$_LOADIDX,
;  1184     1178  6                                 1,
;  1185     1179  6                                 .KEY_DESC [ KEY$B_KEYREF ],
;  1186     1180  6                                 RMS$_DUP );
;  1187     1181  6                !
;  1188     1182  6                ! If this is the first record don't load anything else load the
;  1189     1183  6                ! last record
;  1190     1184  6                !
;  1191     1185  6                IF NOT .CTX [ CTX$V_FST ]
;  1192     1186  6                THEN
;  1193     1187  6                    LOAD_DATA_BUCKET();
;  1194     1188  6                !
;  1195     1189  6                ! Move the key value
;  1196     1190  6                !
;  1197     1191  6                CONV$$COPY_KEY( 4 );
```

```
;  1198          1192  6
;  1199          1193  6              ! Set the sidr array record size
;  1200          1194  6              !
;  1201          1195  6              CTX [ CTX$W_RDS ] = 0;
;  1202          1196  6              !
;  1203          1197  6              ! Set some control fields. NOTE: COPY_KEY sets prologue 3 record
;  1204          1198  6              ! size field NOT counting the pointer so we must add it here
;  1205          1199  6              !
;  1206          1200  6              IF .CONV$GB_PROL_V3
;  1207          1201  6              THEN
;  1208          1202  6                  RECORD_CTRL [ IRC$W_P3SZ ] = .RECORD_CTRL [ IRC$W_P3SZ ] + 7
;  1209          1203  6              ELSE
;  1210          1204  7                  BEGIN
;  1211          1205  7                  !
;  1212          1206  7                  ! Non dup records don't have a dup count
;  1213          1207  7                  !
;  1214          1208  7                  RECORD_CTRL [ IRC$B_CONTROL ] = IRC$M_NOPTRSZ;
;  1215          1209  7                  !
;  1216          1210  7                  RECORD_CTRL [ IRC$W_NODUPSZ ] = .KEY_DESC [ KEY$B_KEYSZ ] + 6
;  1217          1211  7                  !
;  1218          1212  7                  END
;  1219          1213  7                  !
;  1220          1214  5              END;
;  1221          1215  5              !
;  1222          1216  5          ! Load the SIDR array pointer
;  1223          1217  5          !
;  1224          1218  6          BEGIN   ! SIDR local
;  1225          1219  6          !
;  1226          1220  6          DEFINE_VBN_ID_GLOBAL;
;  1227          1221  6          !
;  1228          1222  6          LOCAL   SIDR    : REF BLOCK [ ,BYTE ];
;  1229          1223  6          !
;  1230          1224  6          ! Convert the VBN and the ID that SORT returns in the file
;  1231          1225  6          !
;  1232          1226  6          CONV$$CONVERT_VBN_ID();
;  1233          1227  6          !
;  1234          1228  6          ! Move the record pointer right after the last one, if any
;  1235          1229  6          !
;  1236          1230  6          SIDR = .CTX [ CTX$L_RDP ] + .CTX [ CTX$W_RDS ];
;  1237          1231  6          !
;  1238          1232  6          ! If prologue 3 the ID is bigger
;  1239          1233  6          !
;  1240          1234  6          IF .CONV$GB_PROL_V3
;  1241          1235  6          THEN
;  1242          1236  7              BEGIN
;  1243          1237  7              !
;  1244          1238  7              ! Set the first_key flag if necessary
;  1245          1239  7              !
;  1246          1240  7              IF .DUP
;  1247          1241  7              THEN
;  1248          1242  7                  SIDR [ 0,0,8,0 ] = 2                        ! Can't be first if dup
;  1249          1243  7              ELSE
;  1250          1244  7                  SIDR [ 0,0,8,0 ] = 2 + IRC$M_FIRST_KEY; ! Set flag and size
;  1251          1245  7              !
;  1252          1246  7              SIDR [ 1,0,16,0 ] = .SORT_ID;
;  1253          1247  7              SIDR [ 3,0,32,0 ] = .SORT_VBN;
;  1254          1248  7              CTX [ CTX$W_RDS ] = .CTX [ CTX$W_RDS ] + 7
```

```
;  1255     1249  7            END
;  1256     1250  6        ELSE
;  1257     1251  7            BEGIN
;  1258     1252  7            SIDR [ 0,0,8,0 ] = 2;
;  1259     1253  7            SIDR [ 1,0,8,0 ] = .SORT_ID;
;  1260     1254  7            SIDR [ 2,0,32,0 ] = .SORT_VBN;
;  1261     1255  7            CTX [ CTX$W_RDS ] = .CTX [ CTX$W_RDS ] + 6
;  1262     1256  7            END
;  1263     1257  7
;  1264     1258  5        END;                ! SIDR local
;  1265     1259  5
;  1266     1260  5        ! If we are here then we have processed at least one non null record
;  1267     1261  5        !
;  1268     1262  5        CTX [ CTX$V_FST ] = _CLEAR;
;  1269     1263  5
;  1270     1264  5        ! If this is a non dup key then copy the current record into
;  1271     1265  5        ! dup buffer
;  1272     1266  5        !
;  1273     1267  5        IF NOT .DUP
;  1274     1268  5        THEN
;  1275     1269  5            CH$MOVE( .KEY_DESC [ KEY$B_KEYSZ ],
;  1276     1270  5                     .CONV$GL_RFA_BUFFER + 6,
;  1277     1271  5                     .CONV$GL_DUP_BUF )
;  1278     1272  5
;  1279     1273  5        END                 ! NULL_BLK null key value block
;  1280     1274  5
;  1281     1275  5        END;                ! Main loop
;  1282     1276  5
;  1283     1277  3    ! If we exited because of end of file AND we got at least 1
;  1284     1278  3    !  non-null key value, then finish off the index
;  1285     1279  3    !
;  1286     1280  3    IF .STATUS EQL RMS$_EOF AND NOT .ALL_NULL
;  1287     1281  3    THEN
;  1288     1282  4        BEGIN
;  1289     1283  4
;  1290     1284  4        ! There is a SIDR record left over at this point
;  1291     1285  4        ! We must load it in before we finish off the index
;  1292     1286  4        !
;  1293     1287  4        LOAD_DATA_BUCKET();
;  1294     1288  4
;  1295     1289  4        FINISH_INDEX()
;  1296     1290  4
;  1297     1291  5        END;
;  1298     1292  5
;  1299     1293  3    RETURN
;  1300     1294  3
;  1301     1295  3    END             ! RECORD_CTRL local
;  1302     1296  3
;  1303     1297  1    END;
; INFO#250                L1:1025
; Referenced LOCAL symbol DUP_COUNT is probably not initialized


                                                        .EXTRN  SYS$GET

                                    01FC   8F  BB 00000 CONV$$LOAD_SECONDARY::
```

```
                                                               PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8>            ; 0794
                         5E         14  C2 00004               SUBL2   #20, SP                               ;
                               0000'  CF  B4 00007             CLRW    CONTINUATION                          ; 0860
               0000G  CF 00000000G  8F  D0 0000B               MOVL    #CONV$_WRITEERR, CONV$AB_OUT_RAB+24   ; 0865
                      5A     0000'  CF  D0 00014               MOVL    CONV$GL_CTX_BLOCK, CTX                ; 0869
                      7E        08  AB  9A 00019               MOVZBL  8(KEY_DESC), -(SP)                    ; 0876
                             0000G  30  0001D                  BSBW    CONV$$GET_BUCKET                      ;
                  54  AB        08  AA  D0 00020               MOVL    8(CTX), 84(KEY_DESC)                  ; 0878
                      5A        5C  AA  9E 00025               MOVAB   92(R10), CTX                          ; 0882
                      6E        07  AB  9A 00029               MOVZBL  7(KEY_DESC), (SP)                     ; 0883
                             0000G  30  0002D                  BSBW    CONV$$GET_BUCKET                      ;
                      5E         04  C0 00030                  ADDL2   #4, SP                                ;
                      5A     0000'  CF  D0 00033               MOVL    CONV$GL_CTX_BLOCK, CTX                ; 0885
                      42        10  AB  E9 00038               BLBC    16(KEY_DESC), 4$                      ; 0894
                      29     0000G  CF  E9 0003C               BLBC    CONV$GB_PROL_V3, 3$                   ; 0899
        10        10  AB        03  E1 00041                   BBC     #3, 16(KEY_DESC), 1$                  ; 0904
                      50        2A  AA  3C 00046               MOVZWL  42(CTX), R0                           ; 0912
                      51        14  AB  9A 0004A               MOVZBL  20(KEY_DESC), R1                      ;
                      50            51  C2 0004E               SUBL2   R1, R0                                ;
                      50            04  C2 00051               SUBL2   #4, R0                                ; 0911
                                    0E  11 00054               BRB     2$                                    ; 0912
                      50        2A  AA  3C 00056 1$:           MOVZWL  42(CTX), R0                           ; 0920
                      51        14  AB  9A 0005A               MOVZBL  20(KEY_DESC), R1                      ;
                      50            51  C2 0005E               SUBL2   R1, R0                                ;
                      50            02  C2 00061               SUBL2   #2, R0                                ; 0919
        6E            50            07  C7 00064 2$:           DIVL3   #7, R0, MAX_NUM_DUP                   ; 0920
                                    17  11 00068               BRB     5$                                    ; 0904
                      50        2A  AA  3C 0006A 3$:           MOVZWL  42(CTX), R0                           ; 0929
                      51        14  AB  9A 0006E               MOVZBL  20(KEY_DESC), R1                      ;
                      50            51  C2 00072               SUBL2   R1, R0                                ;
                      50            08  C2 00075               SUBL2   #8, R0                                ; 0928
        6E            50            06  C7 00078               DIVL3   #6, R0, MAX_NUM_DUP                   ; 0929
                                    03  11 0007C               BRB     5$                                    ; 0899
                      6E            01  D0 0007E 4$:           MOVL    #1, MAX_NUM_DUP                       ; 0931
        51            6E            07  C5 00081 5$:           MULL3   #7, MAX_NUM_DUP, R1                   ; 0944
                      50        14  AB  9A 00085               MOVZBL  20(KEY_DESC), R0                      ;
                      50     08 A041  9E 00089                 MOVAB   8(R0)[R1], BYTES                      ;
                                    50  DD 0008E               PUSHL   BYTES                                 ; 0949
                             0000G  30  00090                  BSBW    CONV$$GET_TEMP_VM                     ;
                      5E            04  C0 00093               ADDL2   #4, SP                                ;
                  34  AA            50  D0 00096               MOVL    R0, 52(CTX)                           ;
        50            51        34  AA  C1 0009A               ADDL3   52(CTX), R1, R0                       ; 0953
               0000'  CF        08  A0  9E 0009F               MOVAB   8(R0), CONV$GL_DUP_BUF                ;
                                04  AE  D4 000A5               CLRL    SKIP                                  ; 0968
                      58        30  AA  D0 000A8               MOVL    48(CTX), RECORD_CTRL                  ; 0970
                  08  AE            01  D0 000AC               MOVL    #1, ALL_NULL                          ; 0972
                             0000G  CF  9F 000B0 6$:           PUSHAB  CONV$AB_RFA_RAB                       ; 0977
          00000000G   00            01  FB 000B4               CALLS   #1, SYS$GET                           ;
                  10  AE            50  D0 000BB               MOVL    R0, STATUS                            ;
                      03        10  AE  E8 000BF               BLBS    STATUS, 7$                            ;
                              013E  31 000C3                   BRW     25$                                   ;
                      50     0000G  CF  3C 000C6 7$:           MOVZWL  CONV$AB_RFA_RAB+34, R0                ; 0989
                      50            06  C2 000C9               SUBL2   #6, R0                                ;
  50        14  AB    08            00  ED 000CE               CMPZV   #0, #8, 20(KEY_DESC), R0              ;
                                    DA  1A 000D4               BGTRU   6$                                    ;
        06        10  AB            02  E1 000D6               BBC     #2, 16(KEY_DESC), 8$                  ; 0995
                             0000G  30  000DB                  BSBW    CONV$$CHECK_NULL                      ; 1000
```

```
                        CF              50   E8 000DE                BLBS    R0, 6$                                          ;
                        03        08    AE   E9 000E1 8$:           BLBC    ALL_NULL, 9$                                    ; 1007
                                  08    AE   D4 000E5                CLRL    ALL_NULL                                        ;
                                0000G   30   E8 000E8 9$:           BSBW    CONV$$CHECK_S_DUP                               ; 1011
                        52              50   D0 000EB                MOVL    R0, DUP                                         ;
                        7D        10    AB   E9 000EE                BLBC    16(KEY_DESC), 13$                               ; 1015
                        32              52   E9 000F2                BLBC    DUP, 10$                                        ; 1021
                                  0C    AE   D6 000F5                INCL    DUP_COUNT                                       ; 1025
                        6E        0C    AE   D1 000F8                CMPL    DUP_COUNT, MAX_NUM_DUP                          ; 1030
                                        62   19 000FC                BLSS    12$                                             ;
                                      0000V  30 000FE                BSBW    LOAD_DATA_BUCKET                                ; 1034
                  0000'         CF          01   90 00101           MOVB    #1, DUPLICATE                                   ; 1038
                    04          AE          01   D0 00106           MOVL    #1, SKIP                                        ; 1042
                                            04   DD 0010A           PUSHL   #4                                              ; 1047
                                          0000G  30 0010C           BSBW    CONV$$COPY_KEY                                  ;
                        5E                  04   C0 0010F           ADDL2   #4, SP                                          ;
                                  0C        AE   D4 00112           CLRL    DUP_COUNT                                       ; 1051
                                  3A        AA   B4 00115           CLRW    58(CTX)                                         ; 1055
                        43              0000G    CF E8 00118        BLBS    CONV$GB_PROL_V3, 12$                            ; 1060
                        68                  10   90 0011D           MOVB    #16, (RECORD_CTRL)                              ; 1066
                  02    A8            14    AB   9B 00120           MOVZBW  20(KEY_DESC), 2(RECORD_CTRL)                    ; 1070
                                            39   11 00125           BRB     12$                                             ; 1030
                        18                  6A   E8 00127 10$:      BLBS    (CTX), 11$                                      ; 1081
                                          0000V  30 0012A           BSBW    LOAD_DATA_BUCKET                                ; 1085
                                    0000'   CF   94 0012D           CLRB    DUPLICATE                                       ; 1089
                        0D              04    AE   E9 00131         BLBC    SKIP, 11$                                       ; 1095
                                        04    AE   D4 00135         CLRL    SKIP                                            ; 1098
                                      0000G   30   00138            BSBW    CONV$$WRITE_BUCKET                              ; 1100
                                      0000G   30   00138            BSBW    CONV$$INIT_BUCKET                               ; 1101
                                    0000'     CF   94 0013E         CLRB    CONTINUATION                                    ; 1106
                                              08   DD 00142 11$:    PUSHL   #8                                              ; 1113
                                          0000G  30 00144           BSBW    CONV$$COPY_KEY                                  ;
                        5E                  04   C0 00147           ADDL2   #4, SP                                          ;
                                  0C        AE   D4 0014A           CLRL    DUP_COUNT                                       ; 1117
                                  3A        AA   B4 0014D           CLRW    58(CTX)                                         ; 1121
                        4C              0000G    CF E8 00150        BLBS    CONV$GB_PROL_V3, 16$                            ; 1126
                        68                  01   90 00155           MOVB    #1, (RECORD_CTRL)                               ; 1132
                                  02        A8   D4 00158           CLRL    2(RECORD_CTRL)                                  ; 1136
                            06    A8        14   AB 9B 0015B        MOVZBW  20(KEY_DESC), 6(RECORD_CTRL)                    ; 1140
                        3C              0000G    CF E8 00160 12$:   BLBS    CONV$GB_PROL_V3, 16$                            ; 1148
                45      68                  04   E0 00165           BBS     #4, (RECORD_CTRL), 18$                          ; 1160
                            06    A8            06   A0 00169       ADDW2   #6, 6(RECORD_CTRL)                              ; 1166
                                                43   11 0016D       BRB     19$                                             ; 1148
                        19                      52   E9 0016F 13$:  BLBC    DUP, 14$                                        ; 1175
                                    000184EC    8F   DD 00172       PUSHL   #99564                                          ; 1177
                        7E              15    AB   9A 00178         MOVZBL  21(KEY_DESC), -(SP)                             ; 1179
                                            01   DD 0017C           PUSHL   #1                                              ; 1177
                                  00000000G     8F   DD 0017E       PUSHL   #CONV$_LOADIDX                                  ;
                00000000G       00            04   FB 00184        CALLS   #4, LIB$STOP                                    ;
                                        03          6A   E8 00188 14$:  BLBS    (CTX), 15$                                  ; 1185
                                                  0000V  30 0018E   BSBW    LOAD_DATA_BUCKET                                ; 1187
                                                    04   DD 00191 15$:  PUSHL   #4                                          ; 1191
                                                  0000G  30 00193   BSBW    CONV$$COPY_KEY                                  ;
                                        5E          04   C0 00196   ADDL2   #4, SP                                          ;
                                            3A      AA   B4 00199   CLRW    58(CTX)                                         ; 1195
                                        05      0000G    CF E9 0019C BLBC    CONV$GB_PROL_V3, 17$                            ; 1200
                                        68              07   A0 001A1 16$:  ADDW2   #7, (RECORD_CTRL)                       ; 1202
```

```
                                    0C  11 001A4         BRB     19$                                          ;
                      68            10  90 001A6 17$:    MOVB    #16, (RECORD_CTRL)                           ; 1208
               02     A8       14   AB  9B 001A9         MOVZBW  20(KEY_DESC), 2(RECORD_CTRL)                 ; 1210
               02     A8            06  A0 001AE 18$:    ADDW2   #6, 2(RECORD_CTRL)                           ;
                                 0000G  30 001B2 19$:    BSBW    CONV$$CONVERT_VBN_ID                         ; 1226
                      50       3A   AA  3C 001B5         MOVZWL  58(CTX), SIDR                                ; 1230
                      50       34   AA  C0 001B9         ADDL2   52(CTX), SIDR                                ;
                      1A    0000G   CF  E9 001BD         BLBC    CONV$GB_PROL_V3, 22$                         ; 1234
                      05            52  E9 001C2         BLBC    DUP, 20$                                     ; 1240
                      60            02  90 001C5         MOVB    #2, (SIDR)                                   ; 1242
                                    04  11 001C8         BRB     21$                                          ;
                      60       82   8F  90 001CA 20$:    MOVB    #-126, (SIDR)                                ; 1244
               01     A0            57  B0 001CE 21$:    MOVW    SORT_ID, 1(SIDR)                             ; 1246
               03     A0            56  D0 001D2         MOVL    SORT_VBN, 3(SIDR)                            ; 1247
               3A     AA            07  A0 001D6         ADDW2   #7, 58(CTX)                                  ; 1248
                                    0F  11 001DA         BRB     23$                                          ;
                      60            02  90 001DC 22$:    MOVB    #2, (SIDR)                                   ; 1252
               01     A0            57  90 001DF         MOVB    SORT_ID, 1(SIDR)                             ; 1253
               02     A0            56  D0 001E3         MOVL    SORT_VBN, 2(SIDR)                            ; 1254
               3A     AA            06  A0 001E7         ADDW2   #6, 58(CTX)                                  ; 1255
                      6A            01  8A 001EB 23$:    BICB2   #1, (CTX)                                    ; 1262
                      10            52  E8 001EE         BLBS    DUP, 24$                                     ; 1267
                      51       14   AB  9A 001F1         MOVZBL  20(KEY_DESC), R1                             ; 1269
                      50    0000G   CF  D0 001F5         MOVL    CONV$GL_RFA_BUFFER, R0                       ; 1270
      0000'   DF      06       A0   51  28 001FA         MOVC3   R1, 6(R0), @CONV$GL_DUP_BUF                  ; 1271
                                  FEAC  31 00201 24$:    BRW     6$                                           ; 0979
          0001827A    8F       10   AE  D1 00204 25$:    CMPL    STATUS, #98938                               ; 1280
                                    0A  12 0020C         BNEQ    26$                                          ;
                      06       08   AE  E8 0020E         BLBS    ALL_NULL, 26$                                ;
                                 0000V  30 00212         BSBW    LOAD_DATA_BUCKET                             ; 1287
                                 0000V  30 00215         BSBW    FINISH_INDEX                                 ; 1289
                      5E            14  C0 00218 26$:    ADDL2   #20, SP                                      ; 1297
                           01FC     8F  BA 0021B         POPR    #^M<R2,R3,R4,R5,R6,R7,R8>                    ;
                                    05  05 0021F         RSB                                                  ;

; Routine Size:  544 bytes,    Routine Base:  _CONV$FAST_S + 0238
```

```
;  1305          1298  1 %SBTTL  'LOAD_DATA_BUCKET'
;  1306          1299  1 ROUTINE LOAD_DATA_BUCKET : CL$JSB_REG_8 NOVALUE =
;  1307          1300  1 !++
;  1308          1301  1 !
;  1309          1302  1 ! Functional Description:
;  1310          1303  1 !
;  1311          1304  1 !       Loads a data bucket independent of key of reference in the
;  1312          1305  1 !       index.  On a call to LOAD_DATA_BUCKET a record is loaded into a bucket
;  1313          1306  1 !       and return.  If the record for some reason does not fit into the current
;  1314          1307  1 !       bucket an index is made for the bucket and the bucket is written to the
;  1315          1308  1 !       output file.  The written bucket is initialized and then loaded with
;  1316          1309  1 !       the original record.  The index for a bucket is made by calling
;  1317          1310  1 !       LOAD_INDEX_BUCKET.
;  1318          1311  1 !
;  1319          1312  1 ! Calling Sequence:
;  1320          1313  1 !
;  1321          1314  1 !       LOAD_DATA_BUCKET();
;  1322          1315  1 !
;  1323          1316  1 ! Input Parameters:
;  1324          1317  1 !       none
;  1325          1318  1 !
;  1326          1319  1 ! Implicit Inputs:
;  1327          1320  1 !
;  1328          1321  1 ! Output Parameters:
;  1329          1322  1 !       none
;  1330          1323  1 !
;  1331          1324  1 ! Implicit Outputs:
;  1332          1325  1 !       none
;  1333          1326  1 !
;  1334          1327  1 ! Routine Value:
;  1335          1328  1 !
;  1336          1329  1 !       SS$NORMAL or error codes
;  1337          1330  1 !
;  1338          1331  1 ! Routines Called:
;  1339          1332  1 !
;  1340          1333  1 !       CONV$$GET_BUCKET
;  1341          1334  1 !       LOAD_INDEX_BUCKET
;  1342          1335  1 !       CONV$$SAVE_BUCKET
;  1343          1336  1 !       CONV$$WRITE_BUCKET
;  1344          1337  1 !       CONV$$INIT_BUCKET
;  1345          1338  1 !       CONV$$RESTORE_BUCKET
;  1346          1339  1 !       CONV$$COMPRESS_KEY
;  1347          1340  1 !       CONV$$MAKE_INDEX
;  1348          1341  1 !       CONV$$WRITE_VBN
;  1349          1342  1 !
;  1350          1343  1 ! Side Effects:
;  1351          1344  1 !
;  1352          1345  1 !       Loads a record into a bucket.  Writes buckets and creates indexs
;  1353          1346  1 !       for lower level buckets
;  1354          1347  1 !
;  1355          1348  1 !--
;  1356          1349  1
;  1357          1350  2     BEGIN
;  1358          1351  2
;  1359          1352  2     DEFINE_CTX;
;  1360          1353  2     DEFINE_BUCKET;
;  1361          1354  2     DEFINE_KEY_DESC;
```

```
;  1362     1355  2     DEFINE_RECORD_CTRL;
;  1363     1356  2
;  1364     1357  2     ! Set the bucket pointer to the bucket at this level
;  1365     1358  2     !
;  1366     1359  2     BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  1367     1360  2
;  1368     1361  2     ! Will the record fit into the bucket, if not then call this thing
;  1369     1362  2     ! with an index to the record.
;  1370     1363  2     !
;  1371     1364  2     ! A record will not fit into a bucket if:
;  1372     1365  2     !
;  1373     1366  2     !   For all files:
;  1374     1367  2     !
;  1375     1368  2     ! a) the combined record data size and record control size is greater then
;  1376     1369  2     !    the space availaible in the bucket.
;  1377     1370  2     !
;  1378     1371  2     ! b) the FILL switch is OFF and the space left in the bucket is less then
;  1379     1372  2     !    that allowed by bucket fill quanities
;  1380     1373  2     !
;  1381     1374  2     !   For prologue 1 & 2 files:
;  1382     1375  2     !
;  1383     1376  2     ! c) the record ID of the new record is 0 indicating that the bucket is
;  1384     1377  2     !    filled (as far as id are concerned)
;  1385     1378  2     !
;  1386     1379  4     IF ( ( ( .CTX [ CTX$W_RDS ] + .CTX [ CTX$W_RCS ] ) GTRU
;  1387     1380  4                                                          .CTX [ CTX$W_SPC ] )
;  1388     1381  3          OR
;  1389     1382  3
;  1390     1383  4          ( IF .CONV$GB_PROL_V3
;  1391     1384  4            THEN  0
;  1392     1385  4            ELSE  .BUCKET [ BKT$B_NXTRECID ] EQLU 0 )
;  1393     1386  4
;  1394     1387  3          OR
;  1395     1388  3
;  1396     1389  4          ( ( NOT .CONV$GL_FILL ) AND
;  1397     1390  5            ( LOCAL
;  1398     1391  5                SPACE_USED_IF_RECORD_ADDED;
;  1399     1392  5              SPACE_USED_IF_RECORD_ADDED = .CTX[CTX$W_USE] + .CTX[CTX$W_RCS]
;  1400     1393  5                                                + .CTX[CTX$W_RDS];
;  1401     1394  5                IF .KEY_DESC[KEY$W_DATFILL] - .CTX[CTX$W_USE]                  ! If the difference now
;  1402     1395  5                           LEQ                                                 !      (must be signed)
;  1403     1396  5                   .SPACE_USED_IF_RECORD_ADDED - .KEY_DESC[KEY$W_DATFILL]   ! is less than it would
;  1404     1397  5                THEN                                                           ! be if the record were added,
;  1405     1398  5                    TRUE                                                       ! then don't add it
;  1406     1399  5                ELSE                                                           ! else
;  1407     1400  5                    FALSE                                                      !       go ahead and add it
;  1408     1401  5               ) ) )
;  1409     1402  5
;  1410     1403  2     THEN
;  1411     1404  2         BEGIN   ! Load index block
;  1412     1405  2
;  1413     1406  2         ! If for some reason we dont want to make an index entry for this
;  1414     1407  2         ! record then skip it.
;  1415     1408  2         !
;  1416     1409  2         IF NOT .CONTINUATION
;  1417     1410  3         THEN
;  1418     1411  4             BEGIN
```

```
;  1419          1412  4
;  1420          1413  4               ! Increse the level number for the next index level
;  1421          1414  4               !
;  1422          1415  4               CTX = .CTX + CTX$K_BLN;
;  1423          1416  4
;  1424          1417  4               ! Call to LOAD_INDEX_BUCKET to load the next level of the index
;  1425          1418  4               !
;  1426          1419  4               LOAD_INDEX_BUCKET();
;  1427          1420  4
;  1428          1421  4               ! Return the level
;  1429          1422  4               !
;  1430          1423  4               CTX = .CTX - CTX$K_BLN;
;  1431          1424  4
;  1432          1425  4               ! Restore the bucket pointer to the current level bucket since
;  1433          1426  4               ! we should be looking at some other one.
;  1434          1427  4               !
;  1435          1428  4               BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ]
;  1436          1429  4
;  1437          1430  3               END;
;  1438          1431  3
;  1439          1432  3           ! Write the bucket we filled
;  1440          1433  3           !
;  1441          1434  3           CONV$$WRITE_BUCKET();
;  1442          1435  3
;  1443          1436  3           ! If this is a dup then the next bucket is a continuation bucket
;  1444          1437  3           !
;  1445          1438  3           IF .DUPLICATE
;  1446          1439  3           THEN
;  1447          1440  3               CONTINUATION = _SET
;  1448          1441  3           ELSE
;  1449          1442  3               CONTINUATION = _CLEAR;
;  1450          1443  3
;  1451          1444  3               ! Initialize the bucket to use it again
;  1452          1445  3               !
;  1453          1446  3               CONV$$INIT_BUCKET()
;  1454          1447  3
;  1455          1448  3               END;    ! Load index block
;  1456          1449  2
;  1457          1450  2       BEGIN       ! BKT_*_PTR local
;  1458          1451  2
;  1459          1452  2       ! Load the record into the bucket...
;  1460          1453  2       ! First we must set up pointers to where the record will go in the bucket
;  1461          1454  2       ! These are:
;  1462          1455  2       !
;  1463          1456  2       LOCAL
;  1464          1457  2           BKT_CTRL_PTR,   ! Control information
;  1465          1458  2           BKT_DATA_PTR;   ! Actual data record
;  1466          1459  2
;  1467          1460  2       ! For Prologue 3 files...
;  1468          1461  2       !
;  1469          1462  2       IF .CONV$GB_PROL_V3
;  1470          1463  3       THEN
;  1471          1464  4           BEGIN
;  1472          1465  4
;  1473          1466  4           ! If key compression is on do it
;  1474          1467  4           !
;  1475          1468  4           IF .KEY_DESC [ KEY$V_KEY_COMPR ]
```

```
;  1476     1469  4          THEN
;  1477     1470  4              CONV$$COMPRESS_KEY();
;  1478     1471  4
;  1479     1472  4          ! Key of ref. specific things
;  1480     1473  4          !
;  1481     1474  4          IF .KEY_DESC [ KEY$B_KEYREF ] EQL 0
;  1482     1475  4          THEN
;  1483     1476  4
;  1484     1477  4              ! The Primary key...
;  1485     1478  4              !
;  1486     1479  5              BEGIN
;  1487     1480  5
;  1488     1481  5              ! The record ID
;  1489     1482  5              !
;  1490     1483  5              RECORD_CTRL [ IRC$W_ID ] = .BUCKET [ BKT$W_NXTRECID ];
;  1491     1484  5
;  1492     1485  5              ! The RRV points to it's self ie. it's own ID and VBN
;  1493     1486  5              !
;  1494     1487  5              RECORD_CTRL [ IRC$W_RRV_ID ] = .BUCKET [ BKT$W_NXTRECID ];
;  1495     1488  5              RECORD_CTRL [ IRC$L_RRV_VBN3 ] = .CTX [ CTX$L_CURRENT_VBN ];
;  1496     1489  5
;  1497     1490  5              ! Update the record next record id in the bucket
;  1498     1491  5              !
;  1499     1492  5              BUCKET [ BKT$W_NXTRECID ] = .BUCKET [ BKT$W_NXTRECID ] + 1
;  1500     1493  5
;  1501     1494  5              END
;  1502     1495  4          END
;  1503     1496  3      ELSE
;  1504     1497  3
;  1505     1498  3          ! For prologue 1 and 2 files...
;  1506     1499  3          !
;  1507     1500  4          BEGIN
;  1508     1501  4
;  1509     1502  4          ! The record ID
;  1510     1503  4          !
;  1511     1504  4          RECORD_CTRL [ IRC$B_ID ] = .BUCKET [ BKT$B_NXTRECID ];
;  1512     1505  4
;  1513     1506  4          ! If this is the primary data level the set up the RRV
;  1514     1507  4          !
;  1515     1508  4          IF .KEY_DESC [ KEY$B_KEYREF ] EQL 0
;  1516     1509  4          THEN
;  1517     1510  5              BEGIN
;  1518     1511  5
;  1519     1512  5              ! The RRV points to itself ie. it's own ID and VBN
;  1520     1513  5              !
;  1521     1514  5              RECORD_CTRL [ IRC$B_RRV_ID ] = .BUCKET [ BKT$B_NXTRECID ];
;  1522     1515  5              RECORD_CTRL [ IRC$L_RRV_VBN ] = .CTX [ CTX$L_CURRENT_VBN ]
;  1523     1516  5
;  1524     1517  4              END;
;  1525     1518  4
;  1526     1519  4          ! Update the next record id in the bucket
;  1527     1520  4          !
;  1528     1521  4          BUCKET [ BKT$B_NXTRECID ] = .BUCKET [ BKT$B_NXTRECID ] + 1
;  1529     1522  4
;  1530     1523  3          END;
;  1531     1524  3
;  1532     1525  3      ! For all data levels the control bytes are put at the bucket
```

```
;  1533          1526  3     ! freespace.  The data bytes are put directly after the control.
;  1534          1527  3     !
;  1535          1528  3     BKT_CTRL_PTR = .BUCKET [ BKT$W_FREESPACE ] + .BUCKET;
;  1536          1529  3     BKT_DATA_PTR = .BKT_CTRL_PTR + .CTX [ CTX$W_RCS ];
;  1537          1530  3
;  1538          1531  3     ! Update the bucket pointer (NOTE: Same update for all cases)
;  1539          1532  3     !
;  1540          1533  3     BUCKET [ BKT$W_FREESPACE ] = .BUCKET [ BKT$W_FREESPACE ] +
;  1541          1534  3                                          .CTX [ CTX$W_RCS ] +
;  1542          1535  3                                          .CTX [ CTX$W_RDS ];
;  1543          1536  3
;  1544          1537  3     ! Load the record into the bucket...
;  1545          1538  3     ! Move the control bytes into the bucket
;  1546          1539  3     !
;  1547          1540  3     CH$MOVE( .CTX [ CTX$W_RCS ],.CTX [ CTX$L_RCP ],.BKT_CTRL_PTR );
;  1548          1541  3
;  1549          1542  3     ! Move the data bytes (or sid  array) into the bucket
;  1550          1543  3     !
;  1551          1544  3     CH$MOVE( .CTX [ CTX$W_RDS ],.CTX [ CTX$L_RDP ],.BKT_DATA_PTR );
;  1552          1545  3
;  1553          1546  2     END;        ! BKT_*_PTR local
;  1554          1547  2
;  1555          1548  2     ! Update the amount of space left in the bucket and the amount used
;  1556          1549  2     !
;  1557          1550  3     BEGIN
;  1558          1551  3
;  1559          1552  3     LOCAL
;  1560          1553  3         SPACE_USED;
;  1561          1554  3
;  1562          1555  3     SPACE_USED = .CTX [ CTX$W_RCS ] + .CTX [ CTX$W_RDS ];
;  1563          1556  3
;  1564          1557  3     CTX [ CTX$W_SPC ] = .CTX [ CTX$W_SPC ] - .SPACE_USED;
;  1565          1558  3
;  1566          1559  3     CTX [ CTX$W_USE ] = .CTX [ CTX$W_USE ] + .SPACE_USED;
;  1567          1560  3
;  1568          1561  2     END;
;  1569          1562  2
;  1570          1563  2     ! Make an index for the next level
;  1571          1564  2     !
;  1572          1565  2     CONV$$MAKE_INDEX();
;  1573          1566  2
;  1574          1567  2     ! Set the index record control bytes and bucket pointer
;  1575          1568  2     !
;  1576          1569  2     CONV$$WRITE_VBN();
;  1577          1570  2
;  1578          1571  2     RETURN
;  1579          1572  2
;  1580          1573  1     END;


                                007C   8F   BB 00000 LOAD_DATA_BUCKET:
                                                     PUSHR   #^M<R2,R3,R4,R5,R6>                    ; 1299
                              59         04   AA   D0 00004                MOVL    4(CTX), BUCKET     ; 1359
                              50         3A   AA   3C 00008                MOVZWL  58(CTX), R0        ; 1379
```

```
                                51       38    AA  3C 0000C           MOVZWL  56(CTX), R1                                     ;
                                50            51  C0 00010           ADDL2   R1, R0                                          ;
          50        2A    AA    10            00  ED 00013           CMPZV   #0, #16, 42(CTX), R0                            ; 1380
                                              38  1F 00019           BLSSU   2$                                              ;
                                05     0000G   CF  E8 0001B           BLBS    CONV$GB_PROL_V3, 1$                             ; 1383
                                         06    A9  95 00020           TSTB    6(BUCKET)                                       ; 1385
                                              2E  13 00023           BEQL    2$                                              ;
                                53     0000G   CF  E8 00025 1$:       BLBS    CONV$GL_FILL, 6$                                ; 1389
                                50        2C   AA  3C 0002A           MOVZWL  44(CTX), R0                                     ; 1392
                                51        38   AA  3C 0002E           MOVZWL  56(CTX), R1                                     ;
                                50            51  C0 00032           ADDL2   R1, R0                                          ;
                                52        3A   AA  3C 00035           MOVZWL  58(CTX), R2                                     ; 1393
                                50            52  C0 00039           ADDL2   R2, SPACE_USED_IF_RECORD_ADDED                   ;
                                51        1A   AB  3C 0003C           MOVZWL  26(KEY_DESC), R1                                ; 1394
                                52        2C   AA  3C 00040           MOVZWL  44(CTX), R2                                     ;
                                51            52  C2 00044           SUBL2   R2, R1                                          ;
                                52        1A   AB  3C 00047           MOVZWL  26(KEY_DESC), R2                                ; 1396
                                50            52  C2 0004B           SUBL2   R2, R0                                          ;
                                50            51  D1 0004E           CMPL    R1, R0                                          ;
                                              2A  14 00051           BGTR    6$                                              ;
                                0F     0000'   CF  E8 00053 2$:       BLBS    CONTINUATION, 3$                                ; 1409
                                5A        5C   AA  9E 00058           MOVAB   92(R10), CTX                                    ; 1415
                                         0000V 30  00005C            BSBW    LOAD_INDEX_BUCKET                               ; 1419
                                5A        A4   AA  9E 0005F           MOVAB   -92(R10), CTX                                   ; 1423
                                59        04   AA  D0 00063           MOVL    4(CTX), BUCKET                                  ; 1428
                                         0000G 30  00067 3$:          BSBW    CONV$$WRITE_BUCKET                              ; 1434
                                07     0000'   CF  E9 0006A           BLBC    DUPLICATE, 4$                                   ; 1438
                          0000'   CF            01  90 0006F           MOVB    #1, CONTINUATION                                ; 1440
                                              04  11 00074           BRB     5$                                              ;
                                0000'   CF    94  00076 4$:          CLRB    CONTINUATION                                    ; 1442
                                       0000G  30  0007A 5$:          BSBW    CONV$$INIT_BUCKET                               ; 1446
                                21     0000G   CF  E9 0007D 6$:       BLBC    CONV$GB_PROL_V3, 8$                             ; 1462
                    03    10    AB            06  E1 00082           BBC     #6, 16(KEY_DESC), 7$                            ; 1468
                                       0000G  30  00087           BSBW    CONV$$COMPRESS_KEY                              ; 1470
                                15     AB    95  0008A 7$:          TSTB    21(KEY_DESC)                                    ; 1474
                                              2B  12 0008D           BNEQ    10$                                             ;
                          01    A8        06    A9  B0 0008F           MOVW    6(BUCKET), 1(RECORD_CTRL)                       ; 1483
                          03    A8        06    A9  B0 00094           MOVW    6(BUCKET), 3(RECORD_CTRL)                       ; 1487
                          05    A8        08    AA  D0 00099           MOVL    8(CTX), 5(RECORD_CTRL)                          ; 1488
                                06    A9  B6 0009E           INCW    6(BUCKET)                                               ; 1492
                                              17  11 000A1           BRB     10$                                             ; 1474
                          01    A8        06    A9  90 000A3 8$:       MOVB    6(BUCKET), 1(RECORD_CTRL)                       ; 1504
                                15     AB    95  000A8           TSTB    21(KEY_DESC)                                    ; 1508
                                              0A  12 000AB           BNEQ    9$                                              ;
                          02    A8        06    A9  90 000AD           MOVB    6(BUCKET), 2(RECORD_CTRL)                       ; 1514
                          03    A8        08    AA  D0 000B2           MOVL    8(CTX), 3(RECORD_CTRL)                          ; 1515
                                06    A9  96 000B7 9$:           INCB    6(BUCKET)                                               ; 1521
                                51        04    A9  3C 000BA 10$:      MOVZWL  4(BUCKET), BKT_CTRL_PTR                         ; 1528
                                51            59  C0 000BE           ADDL2   BUCKET, BKT_CTRL_PTR                            ;
                                56        38   AA  3C 000C1           MOVZWL  56(CTX), BKT_DATA_PTR                           ; 1529
                                56            51  C0 000C5           ADDL2   BKT_CTRL_PTR, BKT_DATA_PTR                      ;
                                50        04   A9  3C 000C8           MOVZWL  4(BUCKET), R0                                   ; 1534
                                52        38   AA  3C 000CC           MOVZWL  56(CTX), R2                                     ;
                                50            52  C0 000D0           ADDL2   R2, R0                                          ;
                    04    A9        50    3A  AA  A1 000D3           ADDW3   58(CTX), R0, 4(BUCKET)                          ; 1535
                          61        30    BA  38  AA 28 000D9        MOVC3   56(CTX), @48(CTX), (BKT_CTRL_PTR)               ; 1540
                          66        34    BA  3A  AA 28 000DF        MOVC3   58(CTX), @52(CTX), (BKT_DATA_PTR)               ; 1544
```

```
                                50      38    AA   3C 000E5          MOVZWL  56(CTX), SPACE_USED                     ; 1555
                                51      3A    AA   3C 000E9          MOVZWL  58(CTX), R1                             ;
                                50            51   C0 000ED          ADDL2   R1, SPACE_USED                          ;
                          2A    AA            50   A2 000F0          SUBW2   SPACE_USED, 42(CTX)                     ; 1557
                          2C    AA            50   A0 000F4          ADDW2   SPACE_USED, 44(CTX)                     ; 1559
                                        0000G   30 000F8          BSBW    CONV$$MAKE_INDEX                        ; 1565
                                        0000G   30 000FB          BSBW    CONV$$WRITE_VBN                         ; 1569
                                007C    8F      BA 000FE          POPR    #^M<R2,R3,R4,R5,R6>                     ; 1573
                                                05 00102          RSB                                             ;

; Routine Size:  259 bytes,    Routine Base:  _CONV$FAST_S + 0458
```

```
;  1582     1574  1 %SBTTL  'LOAD_INDEX_BUCKET'
;  1583     1575  1 ROUTINE LOAD_INDEX_BUCKET : CL$JSB_REG_9 NOVALUE =
;  1584     1576  1 !++
;  1585     1577  1 !
;  1586     1578  1 ! Functional Description:
;  1587     1579  1 !
;  1588     1580  1 !       Loads an index bucket independent level in the index.  On a
;  1589     1581  1 !       call to LOAD_INDEX_BUCKET a record is loaded into a bucket and
;  1590     1582  1 !       return.  If the record for some reason does not fit into the current
;  1591     1583  1 !       bucket an index is made for the bucket and the bucket is written to the
;  1592     1584  1 !       output file.  The written bucket is initialized and then loaded with
;  1593     1585  1 !       the original record.  The index for a bucket is made by calling
;  1594     1586  1 !       LOAD_INDEX_BUCKET recursivly. Each recursive call to LOAD_INDEX_BUCKET
;  1595     1587  1 !       is to moving up the index tree.  CTX keeps track to where you are in
;  1596     1588  1 !       the tree.  Most all variables are dependent on CTX so that the
;  1597     1589  1 !       context of each level is saved.
;  1598     1590  1 !
;  1599     1591  1 ! Calling Sequence:
;  1600     1592  1 !
;  1601     1593  1 !       LOAD_INDEX_BUCKET()
;  1602     1594  1 !
;  1603     1595  1 ! Input Parameters:
;  1604     1596  1 !       none
;  1605     1597  1 !
;  1606     1598  1 ! Implicit Inputs:
;  1607     1599  1 !       none
;  1608     1600  1 !
;  1609     1601  1 ! Output Parameters:
;  1610     1602  1 !       none
;  1611     1603  1 !
;  1612     1604  1 ! Implicit Outputs:
;  1613     1605  1 !       none
;  1614     1606  1 !
;  1615     1607  1 ! Routine Value:
;  1616     1608  1 !
;  1617     1609  1 !       SS$NORMAL or error codes
;  1618     1610  1 !
;  1619     1611  1 ! Routines Called:
;  1620     1612  1 !
;  1621     1613  1 !       CONV$$GET_BUCKET
;  1622     1614  1 !       LOAD_INDEX_BUCKET       - Recursive call
;  1623     1615  1 !       CONV$$WRITE_BUCKET
;  1624     1616  1 !       CONV$$INIT_BUCKET
;  1625     1617  1 !       CONV$$COMPRESS_INDEX
;  1626     1618  1 !       CONV$$WRITE_VBN
;  1627     1619  1 !
;  1628     1620  1 ! Side Effects:
;  1629     1621  1 !
;  1630     1622  1 !       Loads a record into a bucket.  Writes buckets and creates indexs
;  1631     1623  1 !       for lower level buckets
;  1632     1624  1 !
;  1633     1625  1 !--
;  1634     1626  1
;  1635     1627  2     BEGIN
;  1636     1628  2
;  1637     1629  2     DEFINE_CTX;
;  1638     1630  2     DEFINE_BUCKET;
```

```
;  1639          1631  2     DEFINE_KEY_DESC;
;  1640          1632  2
;  1641          1633  2     ! Set the bucket pointer to the bucket at this level
;  1642          1634  2     !
;  1643          1635  2     BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  1644          1636  2
;  1645          1637  2     ! See if we have reached the maximun level. (If we have this is the
;  1646          1638  2     ! biggest file in the world!)
;  1647          1639  2     !
;  1648          1640  2     IF .CTX [ CTX$B_LEVEL ] GEQU MAX_IDX_LVL - 1
;  1649          1641  2     THEN
;  1650          1642  2         SIGNAL_STOP( CONV$_IDX_LIM );
;  1651          1643  2
;  1652          1644  2     ! Will the record fit into the bucket, if not then call this thing
;  1653          1645  2     ! with an index to the record.
;  1654          1646  2     !
;  1655          1647  2     ! A record will not fit into a bucket if:
;  1656          1648  2     !
;  1657          1649  2     !   For all files:
;  1658          1650  2     !
;  1659          1651  2     ! a) the combined record data size and record control size is greater then
;  1660          1652  2     !    the space availaible in the bucket.
;  1661          1653  2     !
;  1662          1654  2     ! b) the FILL switch is OFF and the space left in the bucket is less then
;  1663          1655  2     !    that allowed by bucket fill quanities
;  1664          1656  2     !
;  1665          1657  2     !   For prologue 3 files:
;  1666          1658  2     !
;  1667          1659  2     ! c) the bucket below has a different size vbn then this bucket (this
;  1668          1660  2     !    is to keep the same size vbn index buckets)
;  1669          1661  2     !
;  1670          1662  4     IF ( ( ( .CTX [ CTX$W_RDS ] + .CTX [ CTX$W_RCS ] ) GTRU
;  1671          1663  4                                                       .CTX [ CTX$W_SPC ] )
;  1672          1664  3         OR
;  1673          1665  3
;  1674          1666  4         ( ( NOT .CONV$GL_FILL ) AND
;  1675          1667  5           ( LOCAL
;  1676          1668  5               SPACE_USED_IF_RECORD_ADDED;
;  1677          1669  5             SPACE_USED_IF_RECORD_ADDED = .CTX[CTX$W_USE] + .CTX[CTX$W_RCS]
;  1678          1670  5                                                 + .CTX[CTX$W_RDS];
;  1679          1671  5             IF .KEY_DESC[KEY$W_IDXFILL] - .CTX[CTX$W_USE]                 ! If the difference now
;  1680          1672  5                                 LEQ                                       !       (must be signed)
;  1681          1673  5                .SPACE_USED_IF_RECORD_ADDED - .KEY_DESC[KEY$W_IDXFILL]   ! is less than it would
;  1682          1674  5             THEN                                                          ! be if the record were added,
;  1683          1675  5                 TRUE                                                     ! then don't add it
;  1684          1676  5             ELSE                                                          ! else
;  1685          1677  5                 FALSE                                                    !       go ahead and add it
;  1686          1678  4            ) )
;  1687          1679  4
;  1688          1680  3         OR
;  1689          1681  3
;  1690          1682  4         ( IF .CONV$GB_PROL_V3
;  1691          1683  4           THEN
;  1692          1684  5                ( LOCAL CTX_M1 : REF BLOCK [ ,BYTE ];
;  1693          1685  5                  CTX_M1 = .CTX - CTX$K_BLN;
;  1694          1686  5                  IF .BUCKET [ BKT$V_PTR_SZ ] NEQU .CTX_M1 [ CTX$V_VBN ]
;  1695          1687  5                  THEN 1
```

```
;  1696          1688  5                   ELSE 0
;  1697          1689  5                   )
;  1698          1690  5               ELSE 0 ) )
;  1699          1691  5           THEN
;  1700          1692  5               BEGIN   ! Load index block
;  1701          1693  5
;  1702          1694  5               ! Switch for the next index level
;  1703          1695  5               !
;  1704          1696  5               CTX = .CTX + CTX$K_BLN;
;  1705          1697  5
;  1706          1698  5               ! See if the bucket in at the next level is ready if not get it ready
;  1707          1699  5               !
;  1708          1700  5               IF NOT .CTX [ CTX$V_RDY ]
;  1709          1701  5               THEN
;  1710          1702  5
;  1711          1703  5                   ! Get the space for the bucket
;  1712          1704  5                   !
;  1713          1705  5                   CONV$$GET_BUCKET( .KEY_DESC [ KEY$B_IANUM ] );
;  1714          1706  5
;  1715          1707  5               ! Recursive call to LOAD_INDEX_BUCKET to load the next level of the index
;  1716          1708  5               !
;  1717          1709  5               LOAD_INDEX_BUCKET();
;  1718          1710  5
;  1719          1711  5               ! Return the level
;  1720          1712  5               !
;  1721          1713  5               CTX = .CTX - CTX$K_BLN;
;  1722          1714  5
;  1723          1715  5               ! Restore the bucket pointer to the current level bucket since
;  1724          1716  5               ! we should be looking at some other one.
;  1725          1717  5               !
;  1726          1718  5               BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  1727          1719  5
;  1728          1720  5               ! Write the bucket we filled
;  1729          1721  5               !
;  1730          1722  5               CONV$$WRITE_BUCKET();
;  1731          1723  5
;  1732          1724  5               ! Initialize the bucket to use it again
;  1733          1725  5               !
;  1734          1726  5               CONV$$INIT_BUCKET()
;  1735          1727  5
;  1736          1728  2               END;    ! Load index block
;  1737          1729  2
;  1738          1730  3           BEGIN       ! CTX_P1 local
;  1739          1731  3
;  1740          1732  3           LOCAL       CTX_P1  : REF BLOCK [ ,BYTE ];
;  1741          1733  3
;  1742          1734  3           CTX_P1 = .CTX + CTX$K_BLN;
;  1743          1735  3
;  1744          1736  3           ! An index record is made for levels 2 and above ( level 0 and 1 are
;  1745          1737  3           ! made by LOAD_PRIMARY and LOAD_SECONDARY depending on KEY_REF )
;  1746          1738  3           ! NOTE: Do this now because latter the key could get compressed
;  1747          1739  3           !
;  1748          1740  3           CH$MOVE( .CTX [ CTX$W_RDS ],.CTX [ CTX$L_RDP ],.CTX_P1 [ CTX$L_RDP ] );
;  1749          1741  3
;  1750          1742  3           ! Set the size of the data record
;  1751          1743  3           !
;  1752          1744  3           CTX_P1 [ CTX$W_RDS ] = .CTX [ CTX$W_RDS ];
```

```
;  1753     1745  3
;  1754     1746  3      ! Set the size of the control record
;  1755     1747  3      !
;  1756     1748  3      IF .CONV$GB_PROL_V3
;  1757     1749  3      THEN
;  1758     1750  3          CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 2
;  1759     1751  3      ELSE
;  1760     1752  3          CTX_P1 [ CTX$W_RCS ] = .CTX [ CTX$V_VBN ] + 3
;  1761     1753  3
;  1762     1754  3      END;          ! CTX_P1 local
;  1763     1755  3
;  1764     1756  3      BEGIN         ! BKT_*_PTR local
;  1765     1757  3
;  1766     1758  3      ! Load the record into the bucket...
;  1767     1759  3      ! First we must set up pointers to where the record will go in the bucket
;  1768     1760  3      ! These are:
;  1769     1761  3      !
;  1770     1762  3      LOCAL
;  1771     1763  3          BKT_CTRL_PTR,    ! Control information
;  1772     1764  3          BKT_DATA_PTR;    ! Actual data record
;  1773     1765  3
;  1774     1766  3      ! The reason we split them up is because prologue 3 files put the two pieces
;  1775     1767  3      ! in two different places depending on bucket type (ie. INDEX, PRIMARY data
;  1776     1768  3      ! and SECONDARY data bucket.
;  1777     1769  3      !
;  1778     1770  3      ! For Prologue 3 files...
;  1779     1771  3      !
;  1780     1772  3      IF .CONV$GB_PROL_V3
;  1781     1773  3      THEN
;  1782     1774  4          BEGIN
;  1783     1775  4
;  1784     1776  4          ! Prologue 3 files...
;  1785     1777  4          !
;  1786     1778  4          IF .KEY_DESC [ KEY$V_IDX_COMPR ]
;  1787     1779  4          THEN
;  1788     1780  4              CONV$$COMPRESS_INDEX();
;  1789     1781  4
;  1790     1782  4          ! If level 1 save the pointers so we can backup latter
;  1791     1783  4          !
;  1792     1784  4          IF .BUCKET [ BKT$B_LEVEL ] EQLU 1
;  1793     1785  4          THEN
;  1794     1786  5              BEGIN
;  1795     1787  5              SAVE_VBNFS = .BUCKET [ BKT$W_VBNFS ];
;  1796     1788  5              SAVE_KEYFRESPC = .BUCKET [ BKT$W_KEYFRESPC ]
;  1797     1789  4              END;
;  1798     1790  4
;  1799     1791  4          ! Update this pointer first since we go backwards with it
;  1800     1792  4          !
;  1801     1793  4          BUCKET [ BKT$W_VBNFS ] = .BUCKET [ BKT$W_VBNFS ] - .CTX [ CTX$W_RCS ];
;  1802     1794  4
;  1803     1795  4          ! For the index levels the control bytes are put at the bucket
;  1804     1796  4          ! vbn freespace.  The data bytes are put at the key free space.
;  1805     1797  4          !
;  1806     1798  4          BKT_CTRL_PTR = .BUCKET [ BKT$W_VBNFS ] + .BUCKET + 1;
;  1807     1799  4          BKT_DATA_PTR = .BUCKET [ BKT$W_KEYFRESPC ] + .BUCKET;
;  1808     1800  4
;  1809     1801  4          ! Update the rest of the bucket pointers
```

```
;  1810          1802  4          !
;  1811          1803  4          BUCKET [ BKT$W_KEYFRESPC ] = .BUCKET [ BKT$W_KEYFRESPC ] +
;  1812          1804  4                                                   .CTX [ CTX$W_RDS ]
;  1813          1805  4
;  1814          1806  4          END
;  1815          1807  3      ELSE
;  1816          1808  3
;  1817          1809  3          ! For prologue 1 and 2 files...
;  1818          1810  3          !
;  1819          1811  4          BEGIN
;  1820          1812  4
;  1821          1813  4          ! If level 1 save the pointers so we can backup latter
;  1822          1814  4          !
;  1823          1815  4          IF .BUCKET [ BKT$B_LEVEL ] EQLU 1
;  1824          1816  4          THEN
;  1825          1817  4              SAVE_FREESPACE = .BUCKET [ BKT$W_FREESPACE ];
;  1826          1818  4
;  1827          1819  4          ! Set some pointers...
;  1828          1820  4          !
;  1829          1821  4          ! For prologue 1 and 2 files the control bytes are put at the bucket
;  1830          1822  4          ! freespace.  The data bytes are put directly after the control.
;  1831          1823  4          !
;  1832          1824  4          BKT_CTRL_PTR = .BUCKET [ BKT$W_FREESPACE ] + .BUCKET;
;  1833          1825  4          BKT_DATA_PTR = .BKT_CTRL_PTR + .CTX [ CTX$W_RCS ];
;  1834          1826  4
;  1835          1827  4          ! Update the bucket pointer (NOTE: Same update for all cases)
;  1836          1828  4          !
;  1837          1829  4          BUCKET [ BKT$W_FREESPACE ] = .BUCKET [ BKT$W_FREESPACE ] +
;  1838          1830  4                                                   .CTX [ CTX$W_RCS ] +
;  1839          1831  4                                                   .CTX [ CTX$W_RDS ];
;  1840          1832  4
;  1841          1833  3          END;
;  1842          1834
;  1843          1835          ! Load the record into the bucket...
;  1844          1836          ! Move the control bytes into the bucket
;  1845          1837          !
;  1846          1838          CH$MOVE( .CTX [ CTX$W_RCS ],.CTX [ CTX$L_RCP ],.BKT_CTRL_PTR );
;  1847          1839
;  1848          1840          ! Move the data bytes into the bucket
;  1849          1841          !
;  1850          1842          CH$MOVE( .CTX [ CTX$W_RDS ],.CTX [ CTX$L_RDP ],.BKT_DATA_PTR );
;  1851          1843
;  1852          1844          END;        ! BKT_*_PTR local
;  1853          1845
;  1854          1846          ! Update the amount of space left in the bucket and the amount used
;  1855          1847          !
;  1856          1848          BEGIN
;  1857          1849
;  1858          1850          LOCAL
;  1859          1851              SPACE_USED;
;  1860          1852
;  1861          1853          SPACE_USED = .CTX [ CTX$W_RCS ] + .CTX [ CTX$W_RDS ];
;  1862          1854
;  1863          1855          CTX [ CTX$W_SPC ] = .CTX [ CTX$W_SPC ] - .SPACE_USED;
;  1864          1856
;  1865          1857          CTX [ CTX$W_USE ] = .CTX [ CTX$W_USE ] + .SPACE_USED;
;  1866          1858
```

```
;  1867          1859  2     END;
;  1868          1860  2
;  1869          1861  2     ! Set the index record control bytes and bucket pointer
;  1870          1862  2     !
;  1871          1863  2     CONV$$WRITE_VBN();
;  1872          1864  2
;  1873          1865  2     RETURN
;  1874          1866  2
;  1875          1867  1     END;


                                        00FC   8F  BB 00000 LOAD_INDEX_BUCKET:
                                                            PUSHR   #^M<R2,R3,R4,R5,R6,R7>                          ;  1575
                                59        04   AA  D0 00004           MOVL    4(CTX), BUCKET                     ;  1635
                                1F        02   AA  91 00008           CMPB    2(CTX), #31                        ;  1640
                                              OD  1F 0000C           BLSSU   1$
                                    00000000G 8F  DD 0000E           PUSHL   #CONV$_IDX_LIM                     ;  1642
                         00000000G  00        01  FB 00014           CALLS   #1, LIB$STOP
                                50        3A   AA  3C 0001B 1$:      MOVZWL  58(CTX), R0                        ;  1662
                                51        38   AA  3C 0001F           MOVZWL  56(CTX), R1
                                50             51  C0 00023           ADDL2   R1, R0
                50        2A   AA        10        00  ED 00026           CMPZV   #0, #16, 42(CTX), R0           ;  1663
                                              44  1F 0002C           BLSSU   3$
                                29      0000G CF  E8 0002E           BLBS    CONV$GL_FILL, 2$                   ;  1666
                                50        2C   AA  3C 00033           MOVZWL  44(CTX), R0                        ;  1669
                                51        38   AA  3C 00037           MOVZWL  56(CTX), R1
                                50             51  C0 0003B           ADDL2   R1, R0
                                52        3A   AA  3C 0003E           MOVZWL  58(CTX), R2                        ;  1670
                                50             52  C0 00042           ADDL2   R2, SPACE_USED_IF_RECORD_ADDED
                                51        18   AB  3C 00045           MOVZWL  24(KEY_DESC), R1                   ;  1671
                                52        2C   AA  3C 00049           MOVZWL  44(CTX), R2
                                51             52  C2 0004D           SUBL2   R2, R1
                                52        18   AB  3C 00050           MOVZWL  24(KEY_DESC), R2                   ;  1673
                                50             52  C2 00054           SUBL2   R2, R0
                                50             51  D1 00057           CMPL    R1, R0
                                              16  15 0005A           BLEQ    3$
                                34      0000G CF  E9 0005C 2$:       BLBC    CONV$GB_PROL_V3, 5$                ;  1682
                                50        A4   AA  9E 00061           MOVAB   -92(R10), CTX_M1                   ;  1685
                   60           02             05  EF 00065           EXTZV   #5, #2, (CTX_M1), R1               ;  1686
                51 OD   A9      02             03  ED 0006A           CMPZV   #3, #2, 13(BUCKET), R1
                                              23  13 00070           BEQL    5$
                                5A        5C   AA  9E 00072 3$:      MOVAB   92(R10), CTX                       ;  1696
                   0A           6A             02  E0 00076           BBS     #2, (CTX), 4$                      ;  1700
                                7E        06   AB  9A 0007A           MOVZBL  6(KEY_DESC), -(SP)                 ;  1705
                                        0000G 30  0007E           BSBW    CONV$$GET_BUCKET
                                5E             04  C0 00081           ADDL2   #4, SP
                                      FF79     30  00084 4$:      BSBW    LOAD_INDEX_BUCKET                  ;  1709
                                5A        A4   AA  9E 00087           MOVAB   -92(R10), CTX                      ;  1713
                                59        04   AA  D0 0008B           MOVL    4(CTX), BUCKET                     ;  1718
                                        0000G 30  0008F           BSBW    CONV$$WRITE_BUCKET                 ;  1722
                                        0000G 30  00092           BSBW    CONV$$INIT_BUCKET                  ;  1726
                                56        5C   AA  9E 00095 5$:      MOVAB   92(R10), CTX_P1                    ;  1734
                34 B6      34   BA        3A   AA  28 00099           MOVC3   58(CTX), @52(CTX), @52(CTX_P1)     ;  1740
                           3A   A6        3A   AA  B0 000A0           MOVW    58(CTX), 58(CTX_P1)                ;  1744
```

```
                                    0C     0000G  CF   E9 000A5          BLBC    CONV$GB_PROL_V3, 6$                          ; 1748
          50                        6A       02   05   EF 000AA          EXTZV   #5, #2, (CTX), R0                            ; 1750
                38                  A6       50   02   A1 000AF          ADDW3   #2, R0, 56(CTX_P1)
                                                  0A   11 000B4          BRB     7$
          50                        6A       02   05   EF 000B6 6$:      EXTZV   #5, #2, (CTX), R0                            ; 1752
                38                  A6       50   03   A1 000BB          ADDW3   #3, R0, 56(CTX_P1)
                                             49   0000G  CF   E9 000C0 7$:      BLBC    CONV$GB_PROL_V3, 10$                 ; 1772
                      03        10  AB             03   E1 000C5          BBC     #3, 16(KEY_DESC), 8$                         ; 1778
                                           0000G  30 000CA          BSBW    CONV$$COMPRESS_INDEX                         ; 1780
                                       01    0C   A9   91 000CD 8$:      CMPB    12(BUCKET), #1                               ; 1784
                                             13   12 000D1          BNEQ    9$
                                       50    0000'  CF   3C 000D3          MOVZWL  CONV$GW_VBN_FS_PTR, R0                       ; 1787
                                           6049   9F 000D8          PUSHAB  (R0)[BUCKET]
                                  0000'  CF         9E   B0 000DB          MOVW    @(SP)+, SAVE_VBNFS
                                  0000'  CF      04   A9   B0 000E0          MOVW    4(BUCKET), SAVE_KEYFRESPC                    ; 1788
                                       50    0000'  CF   3C 000E6 9$:      MOVZWL  CONV$GW_VBN_FS_PTR, R0                       ; 1793
                                       56      38   AA   3C 000EB          MOVZWL  56(CTX), R6
                                           6049   9F 000EF          PUSHAB  (R0)[BUCKET]
                                       9E        56   A2 000F2          SUBW2   R6, @(SP)+
                                           6049   9F 000F5          PUSHAB  (R0)[BUCKET]                                 ; 1798
                                       50     9E   3C 000F8          MOVZWL  @(SP)+, R0
                                       51   01 A940  9E 000FB          MOVAB   1(BUCKET)[R0], BKT_CTRL_PTR
                                       57      04   A9   3C 00100          MOVZWL  4(BUCKET), BKT_DATA_PTR                      ; 1799
                                       57        59   C0 00104          ADDL2   BUCKET, BKT_DATA_PTR
                                  04   A9      3A   AA   A0 00107          ADDW2   58(CTX), 4(BUCKET)                           ; 1804
                                             25   11 0010C          BRB     12$                                          ; 1803
                                       01    0C   A9   91 0010E 10$:      CMPB    12(BUCKET), #1                               ; 1815
                                             06   12 00112          BNEQ    11$
                                  0000'  CF      04   A9   B0 00114          MOVW    4(BUCKET), SAVE_FREESPACE                    ; 1817
                                       50      04   A9   3C 0011A 11$:      MOVZWL  4(BUCKET), R0                                ; 1824
                    51                 50        59   C1 0011E          ADDL3   BUCKET, R0, BKT_CTRL_PTR
                                       56      38   AA   3C 00122          MOVZWL  56(CTX), R6                                  ; 1825
                    57                 51        56   C1 00126          ADDL3   R6, BKT_CTRL_PTR, BKT_DATA_PTR
                                       50        56   C0 0012A          ADDL2   R6, R0                                       ; 1829
                04   A9                50      3A   AA   A1 0012D          ADDW3   58(CTX), R0, 4(BUCKET)                       ; 1831
                     61        30  BA             56   28 00133 12$:      MOVC3   R6, @48(CTX), (BKT_CTRL_PTR)                ; 1838
                     67        34  BA       3A    AA   28 00138          MOVC3   58(CTX), @52(CTX), (BKT_DATA_PTR)            ; 1842
                                       50      3A   AA   3C 0013E          MOVZWL  58(CTX), SPACE_USED                          ; 1853
                                       50        56   C0 00142          ADDL2   R6, SPACE_USED
                                  2A   AA        50   A2 00145          SUBW2   SPACE_USED, 42(CTX)                          ; 1855
                                  2C   AA        50   A0 00149          ADDW2   SPACE_USED, 44(CTX)                          ; 1857
                                           0000G  30 0014D          BSBW    CONV$$WRITE_VBN                              ; 1863
                                    00FC   8F         BA 00150          POPR    #^M<R2,R3,R4,R5,R6,R7>                       ; 1867
                                                   05 00154          RSB
                                                                                                                           ;

; Routine Size:  341 bytes,    Routine Base:  _CONV$FAST_S + 055B
```

```
;  1877          1868  1 %SBTTL  'FINISH_INDEX'
;  1878          1869  1 ROUTINE FINISH_INDEX : CL$JSB_REG_9 NOVALUE =
;  1879          1870  1 !++
;  1880          1871  1 !
;  1881          1872  1 ! Functional Description:
;  1882          1873  1 !
;  1883          1874  1 !       Loads and writes the last buckets in an index.
;  1884          1875  1 !
;  1885          1876  1 ! Calling Sequence:
;  1886          1877  1 !
;  1887          1878  1 !       FINISH_INDEX()
;  1888          1879  1 !
;  1889          1880  1 ! Input Parameters:
;  1890          1881  1 !       none
;  1891          1882  1 !
;  1892          1883  1 ! Implicit Inputs:
;  1893          1884  1 !       none
;  1894          1885  1 !
;  1895          1886  1 ! Output Parameters:
;  1896          1887  1 !       none
;  1897          1888  1 !
;  1898          1889  1 ! Implicit Outputs:
;  1899          1890  1 !       none
;  1900          1891  1 !
;  1901          1892  1 ! Routine Value:
;  1902          1893  1 !
;  1903          1894  1 !       CONV$_SUCCESS or error codes
;  1904          1895  1 !
;  1905          1896  1 ! Routines Called:
;  1906          1897  1 !
;  1907          1898  1 !       CONV$$WRITE_BUCKET
;  1908          1899  1 !       BACKUP_INDEX
;  1909          1900  1 !       CONV$$CREATE_HIGH_KEY
;  1910          1901  1 !       LOAD_INDEX_BUCKET
;  1911          1902  1 !
;  1912          1903  1 ! Side Effects:
;  1913          1904  1 !
;  1914          1905  1 !       Loads and writes the last buckets in an index.  Deallocates memory used
;  1915          1906  1 !       for bucket buffers.
;  1916          1907  1 !
;  1917          1908  1 !--
;  1918          1909  1
;  1919          1910  2     BEGIN
;  1920          1911  2
;  1921          1912  2     DEFINE_CTX;
;  1922          1913  2     DEFINE_BUCKET;
;  1923          1914  2     DEFINE_KEY_DESC;
;  1924          1915  2
;  1925          1916  2
;  1926          1917  2     ! Finish off the data level bucket.  The reason why we do ths seperatly
;  1927          1918  2     ! is that there are no more records to go in this bucket.  In the index
;  1928          1919  2     ! levels there are.
;  1929          1920  2     !
;  1930          1921  2     CTX = .CONV$GL_CTX_BLOCK;
;  1931          1922  2     BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  1932          1923  2     BUCKET [ BKT$V_LASTBKT ] = _SET;
;  1933          1924  2
```

```
;  1934          1925  2      ! Write the data level bucket
;  1935          1926  2      !
;  1936          1927  2      CONV$$WRITE_BUCKET();
;  1937          1928  2
;  1938          1929  2      ! If the last data bucket was a continuation bucket then backup one
;  1939          1930  2      ! index record an put the high key there
;  1940          1931  2      !
;  1941          1932  2      IF .CONTINUATION
;  1942          1933  2      THEN
;  1943          1934  2          BACKUP_INDEX();
;  1944          1935  2
;  1945          1936  2      ! Create the high key index record to finish things off
;  1946          1937  2      !
;  1947          1938  2      CONV$$CREATE_HIGH_KEY();
;  1948          1939  2
;  1949          1940  2      ! Write the last index records into the buckets and then write the
;  1950          1941  2      ! buckets out
;  1951          1942  2      !
;  1952          1943  2      ! Move up to level 1
;  1953          1944  2      !
;  1954          1945  2      CTX = .CTX + CTX$K_BLN;
;  1955          1946  2
;  1956          1947  2      ! Loop until each level is processed
;  1957          1948  2      !
;  1958          1949  2      WHILE .CTX [ CTX$V_RDY ]
;  1959          1950  2      DO
;  1960          1951  3          BEGIN
;  1961          1952  3
;  1962          1953  3          LOCAL   CTX_P1 : REF BLOCK [ ,BYTE ];
;  1963          1954  3
;  1964          1955  3          ! This call to load bucket will finish off this level bucket and create
;  1965          1956  3          ! the index to the next.
;  1966          1957  3          !
;  1967          1958  3          LOAD_INDEX_BUCKET();
;  1968          1959  3
;  1969          1960  3          ! Before we write out the last bucket set some control info. in it
;  1970          1961  3          !
;  1971          1962  3          BUCKET = .CTX [ CTX$L_CURRENT_BUFFER ];
;  1972          1963  3          BUCKET [ BKT$V_LASTBKT ] = _SET;
;  1973          1964  3
;  1974          1965  3          CTX_P1 = .CTX + CTX$K_BLN;
;  1975          1966  3
;  1976          1967  3          ! If there is no bucket above this one then this is the root
;  1977          1968  3          !
;  1978          1969  4          IF ( NOT .CTX_P1 [ CTX$V_RDY ] )
;  1979          1970  3          THEN
;  1980          1971  4              BEGIN
;  1981          1972  4              BUCKET [ BKT$V_ROOTBKT ] = _SET;
;  1982          1973  4              KEY_DESC [ KEY$B_ROOTLEV ] = .CTX [ CTX$B_LEVEL ];
;  1983          1974  4              KEY_DESC [ KEY$L_ROOTVBN ] = .CTX [ CTX$L_CURRENT_VBN ];
;  1984          1975  4              KEY_DESC [ KEY$V_INITIDX ] = _CLEAR
;  1985          1976  3              END;
;  1986          1977  3
;  1987          1978  3          ! Write the last bucket at this level
;  1988          1979  3          !
;  1989          1980  3          CONV$$WRITE_BUCKET();
;  1990          1981  3
```

```
;  1991          1982  !           ! Clear the bucket ready flag at this level
;  1992          1983  !           !
;  1993          1984  !           CTX [ CTX$V_RDY ] = _CLEAR;
;  1994          1985  !           !
;  1995          1986  !           ! Prepare to work on the bucket one level up
;  1996          1987  !           !
;  1997          1988  !           CTX = .CTX_P1
;  1998          1989  !           !
;  1999          1990  !           END;
;  2000          1991  !       ! Make sure the last IO has completed
;  2001          1992  !       !
;  2002          1993  !       $WAIT( RAB=CONV$AB_OUT_RAB );
;  2003          1994  !       !
;  2004          1995  !       ! Any more IO will be Syncronous
;  2005          1996  !       !
;  2006          1997  !       CONV$AB_OUT_RAB [ RAB$V_ASY ] = _CLEAR;
;  2007          1998  !       !
;  2008          1999  !       RETURN
;  2009          2000  !       !
;  2010          2001  !       END;
;  2011          2002  1       !

                                                        .EXTRN  SYS$WAIT

                                        52  DD 00000 FINISH_INDEX:
                                                    PUSHL   R2                                          ; 1869
                                    5A      0000'   CF  DO 00002                MOVL    CONV$GL_CTX_BLOCK, CTX  ; 1921
                                    59        04    AA  DO 00007                MOVL    4(CTX), BUCKET          ; 1922
                                OD  A9              01  88 0000B                BISB2   #1, 13(BUCKET)          ; 1923
                                        0000G       30  0000F                   BSBW    CONV$$WRITE_BUCKET      ; 1927
                                    03      0000'   CF  E9 00012                BLBC    CONTINUATION, 1$        ; 1932
                                        0000V       30  00017                   BSBW    BACKUP_INDEX            ; 1934
                                        0000G       30  0001A 1$:              BSBW    CONV$$CREATE_HIGH_KEY   ; 1938
                                    5A        5C    AA  9E 0001D                MOVAB   92(R10), CTX            ; 1945
                            30      6A              02  E1 00021 2$:           BBC     #2, (CTX), 4$           ; 1949
                                        FE83        30  00025                   BSBW    LOAD_INDEX_BUCKET       ; 1958
                                    59        04    AA  DO 00028                MOVL    4(CTX), BUCKET          ; 1962
                                OD  A9              01  88 0002C                BISB2   #1, 13(BUCKET)          ; 1963
                                    52        5C    AA  9E 00030                MOVAB   92(R10), CTX_P1         ; 1965
                            12      62              02  E0 00034                BBS     #2, (CTX_P1), 3$        ; 1969
                                OD  A9              02  88 00038                BISB2   #2, 13(BUCKET)          ; 1972
                                09  AB        02    AA  90 0003C                MOVB    2(CTX), 9(KEY_DESC)     ; 1973
                                0C  AB        08    AA  DO 00041                MOVL    8(CTX), 12(KEY_DESC)    ; 1974
                                10  AB              10  8A 00046                BICB2   #16, 16(KEY_DESC)       ; 1975
                                        0000G       30  0004A 3$:              BSBW    CONV$$WRITE_BUCKET      ; 1980
                                    6A              04  8A 0004D                BICB2   #4, (CTX)               ; 1984
                                    5A              52  DO 00050                MOVL    CTX_P1, CTX             ; 1988
                                                    CC  11 00053                BRB     2$
                                        0000G       CF  9F 00055 4$:           PUSHAB  CONV$AB_OUT_RAB         ; 1994
                            00000000G   00          01  FB 00059                CALLS   #1, SYS$WAIT
                                0000G   CF          01  8A 00060                BICB2   #1, CONV$AB_OUT_RAB+4   ; 1998
                                                    04  BA 00065                POPR    #^M<R2>                 ; 2002
                                                    05  00067                   RSB
                                                                                                            ;

; Routine Size:  104 bytes,    Routine Base:  _CONV$FAST_S + 06B0
```

```
;  2013     2003  1 %SBTTL  'BACKUP_INDEX'
;  2014     2004  1 ROUTINE BACKUP_INDEX : CL$JSB_REG_9 NOVALUE =
;  2015     2005  1 !++
;  2016     2006  1 !
;  2017     2007  1 ! Functional Description:
;  2018     2008  1 !
;  2019     2009  1 ! Calling Sequence:
;  2020     2010  1 !
;  2021     2011  1 !       BACKUP_INDEX()
;  2022     2012  1 !
;  2023     2013  1 ! Input Parameters:
;  2024     2014  1 !       none
;  2025     2015  1 !
;  2026     2016  1 ! Implicit Inputs:
;  2027     2017  1 !       none
;  2028     2018  1 !
;  2029     2019  1 ! Output Parameters:
;  2030     2020  1 !       none
;  2031     2021  1 !
;  2032     2022  1 ! Implicit Outputs:
;  2033     2023  1 !       none
;  2034     2024  1 !
;  2035     2025  1 ! Routine Value:
;  2036     2026  1 !       none
;  2037     2027  1 !
;  2038     2028  1 ! Routines Called:
;  2039     2029  1 !       none
;  2040     2030  1 !
;  2041     2031  1 ! Side Effects:
;  2042     2032  1 !
;  2043     2033  1 !       Loads and writes the last buckets in an index.  Deallocates memory used
;  2044     2034  1 !       for bucket buffers.
;  2045     2035  1 !
;  2046     2036  1 !--
;  2047     2037  1
;  2048     2038  2     BEGIN
;  2049     2039  2
;  2050     2040  2     DEFINE_CTX;
;  2051     2041  2     DEFINE_BUCKET;
;  2052     2042  2     DEFINE_KEY_DESC;
;  2053     2043  2
;  2054     2044  2     LOCAL
;  2055     2045  2         VBN_SIZE,
;  2056     2046  2         CTX_P1          : REF BLOCK [ ,BYTE ],
;  2057     2047  2         RECORD_CTRL     : REF BLOCK [ ,BYTE ];
;  2058     2048  2
;  2059     2049  2     CTX_P1 = .CTX + CTX$K_BLN;
;  2060     2050  2
;  2061     2051  2     BUCKET = .CTX_P1 [ CTX$L_CURRENT_BUFFER ];
;  2062     2052  2
;  2063     2053  2     ! If the last data bucket was a continuation bucket then we will be backing
;  2064     2054  2     ! up index record which requires using the vbn in the last record.  We
;  2065     2055  2     ! can fake out conv$$write_vbn (called in conv$$create_high_key) by stuffing
;  2066     2056  2     ! the vbn in the ctx field.  This is ok since it it never referenced again.
;  2067     2057  2     !
;  2068     2058  2     ! Get the size of the vbn in the old record (in bits)
;  2069     2059  2     !
```

```
;  2070          2060  2     VBN_SIZE = ( .CTX_P1 [ CTX$V_VBN ] + 2 ) * 8;
;  2071          2061
;  2072          2062        ! Backup the pointers in the bucket above and get the vbn in the record
;  2073          2063        !
;  2074          2064        IF .CONV$GB_PROL_V3
;  2075          2065        THEN
;  2076          2066            BEGIN
;  2077          2067
;  2078          2068            ! For prologue 3 the vbn is at where we are (they go backwards)
;  2079          2069            !
;  2080          2070            RECORD_CTRL = .BUCKET [ BKT$W_VBNFS ] + .BUCKET + 1;
;  2081          2071
;  2082          2072            CTX [ CTX$L_CURRENT_VBN ] = .RECORD_CTRL [ 0,0,.VBN_SIZE,0 ];
;  2083          2073
;  2084          2074            BUCKET [ BKT$W_VBNFS ] = .SAVE_VBNFS;
;  2085          2075            BUCKET [ BKT$W_KEYFRESPC ] = .SAVE_KEYFRESPC
;  2086          2076
;  2087          2077            END
;  2088          2078        ELSE
;  2089          2079            BEGIN
;  2090          2080
;  2091          2081            BUCKET [ BKT$W_FREESPACE ] = .SAVE_FREESPACE;
;  2092          2082
;  2093          2083            RECORD_CTRL = .BUCKET [ BKT$W_FREESPACE ] + .BUCKET;
;  2094          2084
;  2095          2085            CTX [ CTX$L_CURRENT_VBN ] = .RECORD_CTRL [ 1,0,.VBN_SIZE,0 ]
;  2096          2086
;  2097          2087            END;
;  2098          2088
;  2099          2089        RETURN
;  2100          2090
;  2101          2091  1     END;


                                        52  DD 00000 BACKUP_INDEX:
                                                    PUSHL   R2                                                  ;  2004
                                50       5C   AA  9E 00002          MOVAB   92(R10), CTX_P1                     ;  2049
                                59       04   A0  D0 00006          MOVL    4(CTX_P1), BUCKET                   ;  2051
                50              02            05  EF 0000A          EXTZV   #5, #2, (CTX_P1), R0                ;  2060
                                50            08  C4 0000F          MULL2   #8, VBN_SIZE                        ;
                                50            10  C0 00012          ADDL2   #16, VBN_SIZE                       ;
                                26     0000G  CF  E9 00015          BLBC    CONV$GB_PROL_V3, 1$                 ;  2064
                                52     0000'  CF  3C 0001A          MOVZWL  CONV$GW_VBN_FS_PTR, R2              ;  2070
                                     6249     9F  9F 0001F          PUSHAB  (R2)[BUCKET]                        ;
                                51            9E  3C 00022          MOVZWL  @(SP)+, R1                          ;
                                51       01 A941  9E 00025          MOVAB   1(BUCKET)[R1], RECORD_CTRL          ;
    08  AA          61          50            00  EF 0002A          EXTZV   #0, VBN_SIZE, (RECORD_CTRL), 8(CTX) ;  2072
                                     6249     9F  9F 00030          PUSHAB  (R2)[BUCKET]                        ;  2074
                                9E   0000'    CF  B0 00033          MOVW    SAVE_VBNFS, @(SP)+                  ;
                          04    A9   0000'    CF  B0 00038          MOVW    SAVE_KEYFRESPC, 4(BUCKET)           ;  2075
                                           14  11 0003E          BRB     2$                                  ;
                          04    A9   0000'    CF  B0 00040  1$:    MOVW    SAVE_FREESPACE, 4(BUCKET)           ;  2081
                                51       04   A9  3C 00046          MOVZWL  4(BUCKET), RECORD_CTRL              ;  2083
                                51            59  C0 0004A          ADDL2   BUCKET, RECORD_CTRL                 ;
```

```
      08   AA      01   A1          50          00  EF 0004D          EXTZV   #0, VBN_SIZE, 1(RECORD_CTRL), 8(CTX)      ; 2085
                                                04  BA 00054 2$:      POPR    #^M<R2>                                  ; 2091
                                                    05 00056          RSB                                              ;

; Routine Size:  87 bytes,    Routine Base:  _CONV$FAST_S + 0718

; 2102          2092  1
; 2103          2093  0 END     ELUDOM


                                                                      .EXTRN  LIB$STOP

;                                   PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
;
;  _CONV$FAST_D                   28  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _CONV$FAST_S                 1903  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                                 Library Statistics
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1         18619        56         0        1000       00:01.8
;  _$255$DUA28:[CONV.SRC]CONVERT.L32;1     165        43        26          17       00:00.2


; Information:  1
; Warnings:     0
; Errors:       0


;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:CONVFSTLD/OBJ=OBJ$:CONVFSTLD MSRC$:CONVFSTLD/UPDATE=(ENH$:CONVFSTLD)

; Size:         1903 code + 28 data bytes
; Run Time:       00:43.9
; Elapsed Time:   02:19.3
; Lines/CPU Min:   2863
; Lexemes/CPU-Min: 16797
; Memory Used:  250 pages
; Compilation Complete
```

CONVDATA
LIS

CONVFSTLD
LIS

CONVFSTIO
LIS

CONVFSTRC
LIS

CONVFILES
LIS

CONVERROR
LIS

CONVFASTM
LIS

CONVDEL
LIS